Before using this information and the product that it supports, read the information in “Notices and trademarks” on page 361.
Contents

Tables ........................................ vii

Chapter 1. IBM Initiate Workbench basics .......................... 1
  Using the Update Manager ........................................ 1
  Implementation overview ........................................ 2
  IBM Initiate Workbench concepts and features ................. 2
    Projects .................................................. 2
    Hub Configuration ........................................... 2
    Algorithm Configuration ...................................... 2
    Synchronization and validation ................................ 3
    Deployment .................................................. 3
    Jobs ........................................................ 3
    Custom callout handlers ...................................... 3
  The IBM Initiate Workbench Screen Layout ........................ 4
    Perspectives ............................................... 5
    Views ....................................................... 5
    Editors ..................................................... 8
  Using the Workbench menu options ................................. 9
    Search menu ................................................ 9
    Initiate menu .............................................. 9
  First step in using IBM Initiate Workbench ....................... 10

Chapter 2. Managing projects and Hub connections ............... 11
  Hub connections ............................................. 11
    Adding a connection ........................................ 11
    Editing a connection ........................................ 12
    Removing a connection ...................................... 12
  Managing projects ............................................ 13
    Creating a new project ..................................... 13
    Importing an existing project ............................... 14
    Renaming a project ......................................... 15
    Deleting a project .......................................... 15
  Managing Hub configurations ................................... 15
    Importing a Hub configuration ................................ 15
    Opening a Hub configuration ................................ 16
    Saving a Hub configuration ................................ 17
    Deploying a Hub configuration ................................ 17
  Upgrading Initiate projects .................................... 18

Chapter 3. Jobs and job sets ........................................ 21
  Storage files .................................................. 21
  Data derivation ............................................... 21
  Bulk cross match ............................................. 23
  Creating a Job Set ............................................ 23
    Job set execution .......................................... 24
    Job files and custom Work directories ....................... 25
  Job Categories ............................................... 25
    Algorithm & Analysis jobs .................................. 25
    Bulk Tools jobs ............................................ 39
    Database Tools jobs ........................................ 78
    Full text index jobs ........................................ 82
    Hub Administration jobs ...................................... 83
  Job Set Templates ........................................... 87

Chapter 4. Configuration editor ..................................... 89
  Member types .................................................. 89
    Member types properties ..................................... 89
    Adding a member type ........................................ 90
    Editing a member type ....................................... 90
    Deleting a member type ..................................... 90
  Attributes .................................................... 91
  Entity types .................................................. 93
  Composite Views .............................................. 97
  Sources ...................................................... 101
  Algorithms .................................................... 104
  Attribute types .............................................. 107
    Attribute type properties .................................... 107
    Displaying attribute types .................................. 108
    Adding an attribute type .................................... 108
    Editing an attribute type .................................... 109
    Deleting an attribute type .................................. 110
  History ....................................................... 110
    Member type history ......................................... 111
    Member attribute type history ................................ 111
    Relationship type history ................................... 111
    Relationship attribute type history ........................ 111
  Informational sources .......................................... 111
    Informational source properties ................................ 111
    Adding an informational source ............................. 112
    Editing an informational source ............................. 112
    Deleting an informational source ............................ 112
  Tasks .......................................................... 113
    Task type properties ........................................ 113
    Task status properties ....................................... 114
    Adding a task type .......................................... 115
    Editing a task type ......................................... 115
    Deleting a task type ......................................... 115
    Adding a task status ......................................... 116
    Editing a task status ........................................ 116
    Deleting a task status ....................................... 116
  Linkages ....................................................... 116
    Linkage type properties ..................................... 117
    Linkage status properties .................................... 119
    Editing a linkage type ....................................... 119
    Adding a linkage status ...................................... 120
    Editing a linkage status ..................................... 120
    Deleting a linkage status .................................... 120
  Events ......................................................... 120
    Event properties ............................................ 120
    Adding an event ............................................. 121
    Editing an event ............................................. 121
    Deleting an event ............................................ 121
  Relationship types ............................................ 122
    Relationship type properties ................................ 122
    Relationship type attributes properties ..................... 123
    Relationship type rules properties .......................... 124
    Adding relationship types .................................... 124

© Copyright IBM Corp. 1995, 2012
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editing relationship types</td>
<td>124</td>
</tr>
<tr>
<td>Deleting relationship types</td>
<td>125</td>
</tr>
<tr>
<td>Adding relationship type attributes</td>
<td>125</td>
</tr>
<tr>
<td>Editing relationship type attributes</td>
<td>125</td>
</tr>
<tr>
<td>Deleting relationship type rules</td>
<td>126</td>
</tr>
<tr>
<td>Editing relationship type rules</td>
<td>126</td>
</tr>
<tr>
<td>Deleting relationship type rules</td>
<td>126</td>
</tr>
<tr>
<td>Applications</td>
<td>126</td>
</tr>
<tr>
<td>Application properties</td>
<td>127</td>
</tr>
<tr>
<td>Adding an application</td>
<td>127</td>
</tr>
<tr>
<td>Editing an application</td>
<td>127</td>
</tr>
<tr>
<td>Deleting an application</td>
<td>127</td>
</tr>
<tr>
<td>Application properties properties</td>
<td>128</td>
</tr>
<tr>
<td>Adding application properties</td>
<td>128</td>
</tr>
<tr>
<td>Editing application properties</td>
<td>129</td>
</tr>
<tr>
<td>Deleting application properties</td>
<td>129</td>
</tr>
<tr>
<td>Initiate Web Reports properties</td>
<td>129</td>
</tr>
<tr>
<td>Adding Initiate Web Reports</td>
<td>130</td>
</tr>
<tr>
<td>Handlers</td>
<td>130</td>
</tr>
<tr>
<td>EDT (Enumerated Data Type)</td>
<td>130</td>
</tr>
<tr>
<td>Enumerated data type properties</td>
<td>130</td>
</tr>
<tr>
<td>Adding an EDT code</td>
<td>131</td>
</tr>
<tr>
<td>Editing an EDT code</td>
<td>131</td>
</tr>
<tr>
<td>Deleting an EDT code</td>
<td>131</td>
</tr>
<tr>
<td>Sorting EDT elements</td>
<td>131</td>
</tr>
<tr>
<td>Adding an EDT element</td>
<td>131</td>
</tr>
<tr>
<td>Editing an EDT element</td>
<td>132</td>
</tr>
<tr>
<td>Deleting an EDT element</td>
<td>132</td>
</tr>
<tr>
<td>Strings</td>
<td>132</td>
</tr>
<tr>
<td>String properties</td>
<td>133</td>
</tr>
<tr>
<td>Anonymous addresses</td>
<td>133</td>
</tr>
<tr>
<td>String Examples</td>
<td>133</td>
</tr>
<tr>
<td>String codes</td>
<td>137</td>
</tr>
<tr>
<td>String file fields to be defined</td>
<td>139</td>
</tr>
<tr>
<td>Standardizing anonymous address strings</td>
<td>141</td>
</tr>
<tr>
<td>Anonymous Value Utility</td>
<td>142</td>
</tr>
<tr>
<td>Groups</td>
<td>145</td>
</tr>
<tr>
<td>Running the Synchronize Group Definitions wizard</td>
<td>145</td>
</tr>
<tr>
<td>Group permissions</td>
<td>146</td>
</tr>
<tr>
<td>Auditing view</td>
<td>151</td>
</tr>
<tr>
<td>Setting user audit logging</td>
<td>152</td>
</tr>
<tr>
<td><strong>Chapter 5. Algorithm editor</strong></td>
<td>153</td>
</tr>
<tr>
<td>Sticky selection</td>
<td>153</td>
</tr>
<tr>
<td>Swimlane width</td>
<td>153</td>
</tr>
<tr>
<td>Classic vs. composite algorithm function names</td>
<td>153</td>
</tr>
<tr>
<td>Algorithm and Member Type associations</td>
<td>154</td>
</tr>
<tr>
<td>Loading an algorithm</td>
<td>154</td>
</tr>
<tr>
<td>Saving an algorithm</td>
<td>154</td>
</tr>
<tr>
<td>Renaming an algorithm</td>
<td>155</td>
</tr>
<tr>
<td>Editing an algorithm</td>
<td>155</td>
</tr>
<tr>
<td>The algorithm editor palette</td>
<td>156</td>
</tr>
<tr>
<td>Setting top-level algorithm properties</td>
<td>157</td>
</tr>
<tr>
<td>Top-level algorithm properties</td>
<td>157</td>
</tr>
<tr>
<td>Adding or deleting an attribute</td>
<td>158</td>
</tr>
<tr>
<td>Adding or deleting a standardization function</td>
<td>158</td>
</tr>
<tr>
<td>Setting standardization function properties</td>
<td>159</td>
</tr>
<tr>
<td>Adding or deleting a comparison role</td>
<td>161</td>
</tr>
<tr>
<td>Adding or deleting a query role</td>
<td>162</td>
</tr>
<tr>
<td>Adding or deleting bucketing functions</td>
<td>162</td>
</tr>
<tr>
<td>Adding or deleting comparison functions</td>
<td>163</td>
</tr>
<tr>
<td>Adding or deleting bucketing groups</td>
<td>165</td>
</tr>
<tr>
<td>Connecting a component</td>
<td>166</td>
</tr>
<tr>
<td>Adding tooltips and labels</td>
<td>167</td>
</tr>
<tr>
<td>Weight generation parameters</td>
<td>167</td>
</tr>
<tr>
<td>Weight generation properties</td>
<td>171</td>
</tr>
<tr>
<td>Setting CR and AL thresholds</td>
<td>173</td>
</tr>
<tr>
<td>Setting thresholds manually</td>
<td>173</td>
</tr>
<tr>
<td>Using the Threshold Calculator</td>
<td>173</td>
</tr>
<tr>
<td><strong>Chapter 6. Weight generation</strong></td>
<td>175</td>
</tr>
<tr>
<td>Enabling frequency-based bucketing</td>
<td>176</td>
</tr>
<tr>
<td>Generating weights</td>
<td>176</td>
</tr>
<tr>
<td>Generate Weights process step descriptions</td>
<td>177</td>
</tr>
<tr>
<td>Validating Weights</td>
<td>179</td>
</tr>
<tr>
<td>Validating weights using a job</td>
<td>179</td>
</tr>
<tr>
<td>Validating weights locally</td>
<td>179</td>
</tr>
<tr>
<td>Weight validation rules</td>
<td>180</td>
</tr>
<tr>
<td><strong>Chapter 7. Analytics</strong></td>
<td>187</td>
</tr>
<tr>
<td>The Analytics View</td>
<td>187</td>
</tr>
<tr>
<td>Accessing the analytics view</td>
<td>188</td>
</tr>
<tr>
<td>Connecting to a Hub</td>
<td>188</td>
</tr>
<tr>
<td>Adding a query to a view</td>
<td>189</td>
</tr>
<tr>
<td>Exporting query results to CSV format</td>
<td>189</td>
</tr>
<tr>
<td>Loading previous analysis results</td>
<td>190</td>
</tr>
<tr>
<td>Obtaining data</td>
<td>190</td>
</tr>
<tr>
<td>Preparing analysis data for large datasets</td>
<td>191</td>
</tr>
<tr>
<td>Entity Analysis</td>
<td>191</td>
</tr>
<tr>
<td>Entities By Size</td>
<td>191</td>
</tr>
<tr>
<td>Entity Composition</td>
<td>191</td>
</tr>
<tr>
<td>Entity Size Distribution</td>
<td>192</td>
</tr>
<tr>
<td>Member Comparisons</td>
<td>192</td>
</tr>
<tr>
<td>Member Entity Frequency</td>
<td>193</td>
</tr>
<tr>
<td>Member Entity Values</td>
<td>193</td>
</tr>
<tr>
<td>Members by Entity Count</td>
<td>193</td>
</tr>
<tr>
<td>Score Distribution</td>
<td>193</td>
</tr>
<tr>
<td>Data Analysis</td>
<td>194</td>
</tr>
<tr>
<td>Attribute Completeness</td>
<td>194</td>
</tr>
<tr>
<td>Comparison Data Completeness</td>
<td>194</td>
</tr>
<tr>
<td>Bucket Analysis</td>
<td>194</td>
</tr>
<tr>
<td>Bucket Analysis Overview</td>
<td>194</td>
</tr>
<tr>
<td>Bucket Composition</td>
<td>195</td>
</tr>
<tr>
<td>Bucket Size Distribution</td>
<td>195</td>
</tr>
<tr>
<td>Buckets By Size</td>
<td>195</td>
</tr>
<tr>
<td>Bulk Cross Match Comparison Distribution</td>
<td>196</td>
</tr>
<tr>
<td>Member Bucket Frequency</td>
<td>196</td>
</tr>
<tr>
<td>Member Bucket Values</td>
<td>196</td>
</tr>
<tr>
<td>Member Comparison Distribution</td>
<td>196</td>
</tr>
<tr>
<td>Members By Bucket Count</td>
<td>197</td>
</tr>
<tr>
<td>Linkage Analysis</td>
<td>197</td>
</tr>
<tr>
<td>Member Duplicates</td>
<td>197</td>
</tr>
<tr>
<td>Member Overlaps</td>
<td>197</td>
</tr>
<tr>
<td><strong>Chapter 8. Callout handlers</strong></td>
<td>199</td>
</tr>
<tr>
<td>Custom handlers</td>
<td>199</td>
</tr>
<tr>
<td>Enabling engine callouts</td>
<td>199</td>
</tr>
<tr>
<td>Packaging the handler</td>
<td>200</td>
</tr>
<tr>
<td>Adding external libraries</td>
<td>200</td>
</tr>
</tbody>
</table>
Creating the launch configuration .......... 201
Re-running the Ant script ............... 201
Registering the handler .................. 201
Java handlers ................................ 202
.NET handlers ................................ 202
Handler metadata ............................ 202
Available interaction codes ............... 203
Special considerations for Clover.ETL graphs 203
Adding a handler registration .......... 204
Editing a handler registration ........ 205
Deleting a handler registration ...... 205
Deploying the handler ..................... 205
Testing the handler ....................... 205

Chapter 9. User management .............. 207
Internal LDAP server ....................... 207
Navigating the User Management perspective 209
  Adding users ............................. 209
  Editing users ........................... 210
  Deleting users ........................... 210
  Assigning users to groups ........... 210
  Removing users from groups ....... 211
Navigating the LDAP perspective ...... 211
Connecting to the LDAP server .......... 212
  Importing or exporting LDAP groups and users 212
LDAP groups ................................ 213
  Optional: Create a group DN container 213
  Adding new groups ...................... 213
LDAP users ................................ 214
  Adding new users ...................... 214
  Adding a password that includes multi-byte characters ........... . . . 215
  Adding a user name that includes multi-byte characters .......... 215
  Adding the user to a group ........... 216

Chapter 10. Hub administration ............ 219
Source Sequence Identifiers ................ 219

Chapter 11. Operational monitoring ....... 221
Accessing the JMX browser ............... 221
  The MBean Explorer view .............. 221
  Listeners node .......................... 222
  Log4j node ................................ 223
  Search node ............................. 223
  ThreadPools node ....................... 225
  Administrative actions ............... 226

Chapter 12. IBM Initiate Pair Manager .. 227
The sample pair file ....................... 227
Using the IBM Initiate Pair Manager ...... 228
  Filtering members by status ........ 228
  Selecting data columns to display .... 229
  Handling large sample pair files .... 229

Chapter 13. Initiate Inspector
Configuration ................................ 231
Implementing configuration changes ...... 231
IBM Initiate Inspector Configuration Editor ...... 231
Attribute Display page .................... 232
Custom Task Summary Page .............. 233
General Preferences page ....... 234
Member and Entity page .............. 236
Search Forms page ....................... 237
Search Results page ..................... 238

Chapter 14. IBM Initiate Flexible Search ......... 241
Introduction ................................ 241
Creating an index configuration .......... 241
Configuring the search index .......... 242
Index configuration file .............. 243
  Index fields ........................... 243
  General configuration ............... 245
  Search tuning ......................... 245

Appendix A. Algorithm function
descriptions ................................ 247
Standardization functions ................ 250
  Anonymous values ....................... 251
  Equivalent values ....................... 251
  Address standardization functions (street, postal, ZIP code, and email) 252
  Attribute standardization functions ........ 264
  Biometric standardization functions ........ 266
  Date standardization functions ........ 268
  Geographic standardization functions .......... 271
  Identifier standardization functions .......... 273
  Name standardization functions .......... 274
  Phone standardization functions .......... 285
  Miscellaneous standardization functions .......... 288
Comparison functions ..................... 289
  Token comparison ....................... 290
  Address comparison functions .......... 291
  Attribute comparison functions .......... 300
  Biometric comparison functions .......... 301
  Date comparison functions ................ 304
  Edit distance comparison functions .......... 310
  Equivalent comparison functions .......... 315
  False Positive comparison functions .......... 315
  Geographic comparison functions .......... 317
  Name comparison functions ............ 318
Bucketing .................................... 327
  Bucket functions ....................... 328
  Generation Functions ................... 328

Appendix B. Algorithm validation rule
syntax .................................. 335
Adding validation rules .................. 335
Standardization function rules .......... 335
Comparison function rules .............. 336
Bucketing Functions ............... 336
Generation Functions .................. 336
Sample Validation Rules XML .......... 337

Appendix C. Understanding weight
table names .................................. 345
Tables

1. Compare Members (mpimcomp) options 26
2. Generate Frequency Stats (mpxfreq) options 27
3. Generate Threshold Analysis Pairs options 31
4. Generate weights options 33
5. Bucket analysis preparation options 38
6. Validate weights options 38
7. Compare Members in Bulk (mpxcomp) options 42
8. Derive Data and Create UNLs (mpxdata) options 49
9. Derive Data from Hub (mpxredvd) options 56
10. Derive Data from UNLs (mpxfsdvd) options 59
11. Export Linkage History (mpxxeia) options 63
12. Link Entities (mpxlink) options 65
13. Steps for executing Link Relationships 72
14. Link Relationships options 72
15. Prepare Binary Files (mpxprep) options 73
16. Sort Binary Link Data (mpxsort) options 76
17. Load UNLs to DB (madunlload) options 78
18. Unload UNLs from DB (madunlunload) options 81
19. Index by Members options 82
20. Index by UNLs options 83
21. Deploy Hub Configuration options 84
22. Execute Remote Graph options 85
23. Get File options 87
24. Member types properties 89
25. Attributes properties 91
26. Entity type properties 93
27. Source priorities for entity types 96
28. Composite view properties 98
29. Sources properties 102
30. Attribute type properties 107
31. Attribute field properties 108
32. Informational source properties 112
33. Task type properties 114
34. Task status properties 114
35. Linkage type properties 117
36. Linkage status properties 119
37. Event properties 120
38. Relationship type properties 122
39. Relationship type attributes properties 123
40. Relationship type rules properties 124
41. Application properties 127
42. Application Properties properties 128
43. Web Reports properties 129
44. Enumerated Data Type properties 131
45. String properties 133
46. String file fields to be defined 139
47. Interaction guidelines 147
48. Matched pair threshold 167
49. Attribute matched pair threshold 168
50. Attribute matched pair percentage threshold 168
51. Data quality percentage for initial weight estimates 169
52. Minimum attribute count 169
53. Convergence threshold 169
54. False negative rate 169
55. False positive rate 170
56. Weight table percentage cut-off 170
57. Minimum weight frequency 170
58. Weight generation utilities used for various system states 171
59. Weight generation properties with prompts 172
60. Weight generation properties without prompts 172
61. Available Queries List 187
62. Analytics view toolbar icons 188
63. Available interaction codes 203
64. Clover handler metadata 204
65. Preconfigured LDAP groups 208
66. Interactions in the monitoring perspective 222
67. Composite-to-classic standardization function Map 247
68. Composite-to-classic comparison function mapping 248
69. Composite-to-classic bucketing function mapping 249
70. Weight tables and corresponding wgttypes 345
71. Initiate tools executable through cheat sheets 350
72. Sample mpxdata configuration file 354
73. IBM resources 371
74. Providing feedback to IBM 371
Chapter 1. IBM Initiate Workbench basics

The IBM® Initiate® Workbench application enables implementers and administrators to easily manage the IBM Initiate Master Data Service® software environment. Use this application to maintain user security, create composite views, edit data dictionary tables, and to develop member logical models, flows, and mappings to data sources. This guide is designed as a resource for system, application, and database administrators and those responsible for implementing Initiate software.

IBM Initiate Workbench release 9.7 is an Eclipse-based technology and runs on computers using Microsoft Windows. Its primary use is to configure the Master Data Engine to operate as a Data Hub, but it has many additional features. This section describes how to start IBM Initiate Workbench and use its menus and views. For information on installing IBM Initiate Workbench, refer to the IBM Initiate Workbench Installation Guide. For information on Eclipse, refer to http://www.eclipse.org.

Note: The Sun Java JDK is installed with IBM Initiate Workbench and is required to run this application.

From the Microsoft Windows start menu select:

Start > All Programs > IBM Initiate > Workbench 9.7.x

In order for the graphical interface to be displayed properly in Windows, your screen resolution setting should be 1024 x 768 pixels or higher.

You will first be prompted to enter the path for your workspace. You may accept the default, located at windows_user_profile\workspace, or type the desired path. Throughout this documentation, this path will be referred to as workspace, for example workspace\project_name\jobTemplates might refer to C:\Documents and Settings\jdoe\workspace\localhub\jobTemplates. To always use this workspace location, check the **Use this as the default and do not ask again** box. IBM Initiate Workbench opens.

Using the Update Manager

About this task

To use the Update Manager built into IBM Initiate Workbench, follow the instructions below.

Procedure

1. From the main menu, select **Window > Preferences > General > Capabilities**.
2. Enable the **Classic Update** option.
3. Use the **Help > Software Updates > Find and Install** or **Help > Software Updates > Manage Configurations** option to search for software updates.
Implementation overview

The steps below outline the high-level implementation process for deploying the IBM Initiate Master Data Service platform. Items 2, 3, 4, 6, 7, 8, 9, and 11 are performed using IBM Initiate Workbench.

1. Review customer requirements.
2. Configure the member model (see Chapter 4, “Configuration editor,” on page 89).
5. Obtain the data extract.
7. Derive the data (see Chapter 3, “Jobs and job sets,” on page 21).
8. Generate weights (see Chapter 6, “Weight generation,” on page 175).
10. Analyze and review.
11. Reconfigure algorithm and repeat remaining steps as needed.

As you can see, IBM Initiate Workbench is used extensively in the implementation process. Instructions for using the IBM Initiate Workbench user interface in performing these steps are discussed in this chapter.

IBM Initiate Workbench concepts and features

IBM Initiate Workbench enables you to create and/or edit a Master Data Engine configuration locally and then remotely deploy the validated configuration to a Master Data Engine instance. The following sections describe the primary features of IBM Initiate Workbench.

Projects

A IBM Initiate Workbench project is used to organize a Hub configuration that will be deployed to a Master Data Engine instance. Multiple projects can be worked on for the same Hub or multiple Hubs. Projects are persisted locally in directories below the workspace directory established the first time IBM Initiate Workbench is started. Projects are visible in the Navigator window in IBM Initiate Workbench.

Hub Configuration

A Hub configuration is a view of the members, attributes, and segments defined for an Initiate software implementation. This data dictionary configuration is stored in the Initiate Member Model file (.imm) found in the Navigator pane. The file is given the same name as the project, and additional information is populated based on the template option you chose during the project creation process. Details on configuring the Hub are found in Chapter 4, “Configuration editor,” on page 89.

Algorithm Configuration

Algorithms are used within the Hub to compare and score member attribute similarities and differences. Initiate software applies the algorithms to data to create tasks and to support search functionality. The algorithms are tailored to use...
attributes specific to your business, and are defined using the Algorithm Editor. Details on configuring algorithms are found in Chapter 5, “Algorithm editor,” on page 153.

Synchronization and validation
IBM Initiate Workbench also provides advanced synchronization and validation features that help keep change management to a minimum and prevent invalid configurations from reaching the Hub.

Whenever a project resource is modified, IBM Initiate Workbench works behind the scenes to ensure that all parts of the metadata (models, algorithms, and so forth) remain in synchronization. When a change to one resource adversely affects another resource, a task (or set of tasks) is generated in the Problems view. This view enables you to track each new problem the resource has generated. The tasks can then be worked in any order to resolve the issue.

Typographical errors, incorrect syntax, and so forth are all checked and validated. Any problems encountered are entered as tasks in the Problems view for tracking and correction.

Validation tasks in the Problems view
When selecting tasks in the Problems view, the task opens the appropriate editor and automatically selects the corresponding problem area (for example, a comparison role or string code). Depending on the task in question, the task is either immediately removed when the invalid issue is corrected, or it is removed the next time a validation check is run (when a project resource is saved).

Each IBM Initiate Workbench project is validated when it is opened (or after restarting IBM Initiate Workbench). This guarantees that the project resources are validated against any external changes made since the project was last edited.

Some tasks have associated “Quick Fixes” which are available for guided resolution of the problem. To access the “Quick Fix” menu: select the task and bring up the context menu by right-clicking on it, then select the “Quick Fix” option. If the option is not selectable then there is no quick fix for that problem, and you will have to correct it without guided assistance.

Deployment
Hub configurations and algorithms are deployed directly to the Master Data Engine instance. IBM Initiate Workbench includes a wizard that takes you through the steps of deploying. See “Deploying a Hub configuration” on page 17.

Jobs
Some tasks need to be performed directly on the Hub, outside of configuration deployment. IBM Initiate Workbench provides a means for performing single jobs or grouping jobs within a job set, executing them directly on the Hub, and displaying the progress or state of the job execution within a IBM Initiate Workbench view. In many cases, job results can be retrieved or viewed from the Hub by right-clicking the completed job and selecting Get Job Results. Details on executing jobs are found in Chapter 3, “Jobs and job sets,” on page 21.

Custom callout handlers
IBM Initiate Workbench enables you to register custom callout handlers that manage your special business needs, such as:
Callouts to a third-party API
Apply conditional security for certain types of interactions
Send event notifications to an external message queue, e-mail address, file system, and so forth.

Consult the IBM Initiate Master Data Service SDK Reference for Java and Web Services for more information on creating custom handlers. Refer to Chapter 8, "Callout handlers," on page 199 for information on registering and deploying the handlers.

Undo/Redo
IBM Initiate Workbench provides automatic local history undo/redo and compare/replace across save boundaries and Workbench instances. This history does not jeopardize the integrity of existing artifacts and is available for any text-based file, including XML and Java. Once a project is saved, however, the undo/redo history is cleared.

Versioning and source control
Workbench is an Eclipse-based technology and accepts many source control plug-ins. All project resources are text-based files which can be stored and versioned easily into source control systems. Refer to www.eclipse.org and search for source control plug-ins.

Cheat sheets
As an Eclipse-based application, Workbench supports the use of cheat sheets to guide users through completing specific tasks. To launch a cheat sheet, go to Help > Cheat Sheets.... If this command is not in the menu, you can add it by opening Window > Customize Perspective > Commands, and checking Cheat Sheets.

Workbench comes with a Tuning Search cheat sheet out-of-the-box, and more can be created on an as-needed basis. Instructions for creating cheat sheets and accessing Workbench utilities via cheat sheets can be found in Appendix E, "Creating cheat sheets for Initiate tools," on page 349.

Log in
Many IBM Initiate Workbench features can be used while disconnected from a Hub. Some, however, require you to log in to a Hub. Once you log in, a dialog opens to display the date and timestamp of your last successful login to the selected Hub. The last successful login is not application-specific. If you last logged into the Hub from IBM Initiate Inspector, for example, that login timestamp is displayed. You can enable or disable the Show login confirmation dialog option by selecting Window > Preferences > Initiate.

The default system administrator login name and password is system / system but Initiate Systems, Inc. suggests you change this at your earliest convenience. Procedures for adding, editing and deleting users are described in Chapter 9, “User management,” on page 207.

The IBM Initiate Workbench Screen Layout
The IBM Initiate Workbench screen utilizes the most common features of the Eclipse development tool, including perspectives, views and editors.
Perspectives

The Workbench user interface contains an editor and a set of work areas called views. Together, they form a Workbench perspective. This perspective defines an optimized, task-based orientation of the most-used functions of Workbench.

The shortcut bar appears along the right edge of the window and provides quick access to all perspectives currently in use. It also contains any fast views you create as shortcuts to the views you use most often. The Configuration, LDAP, User Management, CloverETL and Analytics perspectives are the perspectives most used within Workbench.

Note: To change the docking, right-click on the perspective toolbar and select your desired docking location. The toolbar is docked at the Top Left position by default.

The perspectives that enable you to work with the Hub and Hub data are:

- **Analytics**: This perspective displays four Analytics views by default, which enable you to view multiple analytics queries side-by-side. You can find more information about using the views in this perspective in Chapter 7, “Analytics,” on page 187.

- **Configuration**: The main editor window enables you to manage the dictionary data for the Hub or to design and edit algorithms. You can find more information about using the views in this perspective in Chapter 4, “Configuration editor,” on page 89 and Chapter 5, “Algorithm editor,” on page 153.

- **Clover ETL**: The CloverETL function lets you design and execute data extract, transfer and load (ETL) operations in a graph format. You can find more information about using the views in this perspective in the IBM Initiate Workbench Clover User’s Guide.

- **Inspector Configuration**: The Inspector Configuration perspective is used to edit the inspector.arm file, which stores the IBM Initiate Inspector application configuration. When changes to this configuration file are deployed to the Hub, any browser used to open the IBM Initiate Inspector application automatically receives the IBM Initiate Inspector configuration when the user logs in. Details on using the views in the Inspector Configuration perspective are located in Chapter 13, “Initiate Inspector Configuration,” on page 231.

- **LDAP**: The main editor window displays the LDAP server information for editing as needed. You can find more information about using the views in this perspective in Chapter 9, “User management,” on page 207.

- **Monitoring**: The JMX browser packaged with Workbench provides an easy method for monitoring performance and resource consumption. You can find more information about using the views in this perspective in Chapter 11, “Operational monitoring,” on page 221.

- **User management**: This perspective enables those unfamiliar with LDAP to add users and assign them to groups. You can find more information about using the views in this perspective in Chapter 9, “User management,” on page 207.

Views

Views are individual windows that contain specific types of data. Most views can be moved to different areas of the screen by dragging and dropping their tabs. To change views, select the **Window** menu and click **Show View**. You can then choose from a variety of views.
The following is a brief description of the different views included in the Workbench user interface.

**Navigator view**
The Navigator view appears at the upper left-hand side of the window and provides a tree structure for browsing the Workbench artifacts. The root of the tree structure begins at the workspace folder in your Master Data Engine software installation directory.

From the Navigator view, you can:
- Traverse project directories
- Open and view project files
- Copy, paste, move, delete and rename project files
- Select a working set of files (and hide files not used in the working set)
- Deselect a working set of files
- Compare files with the local history
- Compare a file with another file
- Replace a file with one from the local history
- Replace a file with another file

If the Navigator view is hidden, you can display it by selecting to open the Window -> Show View > Other to open the Show View dialog, then expanding General and double-clicking Navigator.

**Properties view**
The Properties view appears at the bottom left of the window and enables you to edit the property values of any component you create.

If the Properties view is hidden, you can display it by selecting to open the Window -> Show View > Other to open the Show View dialog, then expanding General and double-clicking Properties.

**Problems view**
The Problems view appears at the bottom left of the window and provides a list of configuration and validation problems in Workbench. Most validations are done when file resources in the project are saved, so errors appear instantly. Double-click a problem in the Problems view to go to the item that generated the error. See “Validation tasks in the Problems view” on page 3.

If the Problems view is hidden, you can display it by selecting to open the Window -> Show View > Other to open the Show View dialog, then expanding General and double-clicking Problems.

**Console view**
A Console view appears at the bottom of the window and shows progress messages and errors during extensive background tasks.

If the Console view is hidden, you can display it by selecting Window -> Show View -> Other to open the Show View dialog, then expanding General and double-clicking Console.
**Jobs view**
The Jobs view appears at the bottom right of the window and shows progress or completion status for job sets executed from the jobs wizard. Click the + icon to expand the job and view output details, and click the - icon to collapse it.

If the Jobs view is hidden, you can display it by selecting to open the Window -> Show View > Other to open the Show View dialog, then expanding Workbench and double-clicking Jobs.

**Analytics view**
The Analytics view appears at the bottom right of the window and displays the results of an analytics query. You need to create a connection to the Hub and then specify a query to run in order to see data in this view.

If the Analytics view is hidden, you can display it by selecting to open the Window -> Show View > Other to open the Show View dialog, then expanding Workbench and double-clicking Analytics.

**Search view**
The Search view appears at the bottom right of the window and displays the results of a Search. You can double-click a row in the Search view to open the corresponding configuration object.

If the Search view is hidden, you can display it by selecting to open the Window -> Show View > Other to open the Show View dialog, then expanding General and double-clicking Search.

**MBean Explorer view**
The MBean Explorer view appears at the left of the window in the Monitoring perspective and displays the MBeans accessible on the connected Hub. You can double-click a row in the MBean Explorer view to open the corresponding MBean object.

If the MBean Explorer view is hidden, you can display it by selecting to open the Window -> Show View > Other to open the Show View dialog, then expanding Monitoring and double-clicking MBean Explorer. You can also display the view by clicking the MBean Explorer button in the bottom left corner of the Monitoring perspective.

**Configuration comparison view**
When changes have been made to a Hub configuration or algorithm in Workbench but have not yet been deployed to the Hub, you can view the differences between the configuration on the Hub and the one in Workbench by using the Configuration Comparison view.

Similarly, if the Hub configuration was updated using a different Workbench instance, you can compare your current configuration with the changes made to the Hub.

**Comparing configurations:**
**Procedure**
1. Access the view by selecting the Window > Show View > Other... menu option and then selecting Configuration Comparison in the Workbench group. The Configuration Comparison view opens.
2. Click the source selection icon in the upper right corner of the view to open the Configuration Comparison Source Selection dialog.
3. For each of the Sources, select either a project or a Hub. You can compare two projects, two Hubs or a Hub and a project. When selecting a project, you can optionally include any unsaved editor changes by enabling the Include unsaved editor changes checkbox.

4. Click OK. The Configuration Comparison view is populated with comparison data. Rows highlighted in yellow indicate that one or more of its child objects differ between the two sources. Rows highlighted in pink indicate the differing object(s).
   a. To view algorithm differences, click the Algorithms tab.
   b. To view Hub configuration differences, click the Hub Configuration tab.
   c. To show only the objects that differ, enable the Only show differences checkbox.

5. If one of the Hub configurations or algorithms change, repeat steps 2-4 to refresh the view.

Editors

The main portion of the IBM Initiate Workbench perspectives is allocated to editors. IBM Initiate Workbench provides several special types of editors, such as the Hub configuration editor, algorithm editor and IBM Initiate Inspector configuration editor. The Hub configuration editor is for working with .imm files, the algorithm editor is for working with .alg files, and the Inspector Configuration editor is for working with .arm files. IBM Initiate Workbench also supports other editor types, including standard text and Java editors.

About the Hub configuration editor

The Hub configuration editor shows a view of member types, entity types, composite views, sources, attribute types, strings, security and so on. The following is a typical representation of a Hub configuration imported into Workbench. Refer to Chapter 4, “Configuration editor,” on page 89 for more information on using the Hub configuration editor.

Most of the tables in the Hub configuration editor can be sorted by clicking on a column heading. A small arrow indicates how the data is sorted: an up arrow mean the sort is ascending, and a down arrow means the sort is descending. Note that not all tables or columns can be sorted.

About the Algorithm editor

The Algorithm Editor enables you to edit the algorithm files which are used by the Hub to apply comparison logic. Refer to Chapter 5, “Algorithm editor,” on page 153 for more information on using this editor.

About the Inspector Configuration editor

The Inspector Configuration Editor enables you to edit the inspector configuration file. Once the file is deployed to the Hub, all users of IBM Initiate Inspector will have access to the same configuration. The editor can be accessed by double-clicking the inspector.arm file in the inspector folder within the Navigator. To create a new inspector.arm file, select Initiate > Create Default Inspector Configuration... from the main menu. Refer to Chapter 13, “Initiate Inspector Configuration,” on page 231 for more information on using the Inspector Configuration editor.

About the Flexible Search configuration editor

The IBM Initiate Flexible Search editor enables you to edit the search index configuration file. Once the file is deployed to the Hub, all users of IBM Initiate Flexible Search will have access to the same index configuration.
The editor can be accessed by double-clicking the flexsearch.fsm file in the flexsearch folder within the Navigator. To create a new flexsearch.fsm file, select **Initiate > Create Default Flexible Search Configuration...** from the main menu. Refer to “Creating an index configuration” on page 241 or more information on using the Flexible Search Configuration editor.

---

### Using the Workbench menu options

When you start Workbench, you are presented with the basic platform view with a full-featured menu at the top. Most of the menus are standard Eclipse menus. The following section describes the operations you can perform using the Search and Initiate menu options. For information on using the other menus, please consult the Eclipse documentation or help.

#### Search menu

The Search menu enables you to search for text strings in files. You can also search for objects within a Hub configuration using a text search. Use the Hub Configuration Search tab to search for objects whose identifying attributes include the specified text string. The search results are displayed in the Search view (see “Search view” on page 7).

#### Initiate menu

The Initiate menu provides a way to perform the following operations.

- Create a new Initiate project (Chapter 2, “Managing projects and Hub connections,” on page 11)
- Import a Hub configuration (Chapter 2, “Managing projects and Hub connections,” on page 11)
- Create a Flexible Search configuration file. (refer to “Creating an index configuration” on page 241)
- Create a default Inspector configuration file (Chapter 13, “Initiate Inspector Configuration,” on page 231)
- Generate Enterprise Services (refer to the IBM Initiate Enterprise Service Oriented Architecture (SOA) Toolkit User’s Guide)
- Deploy handlers (Chapter 8, “Callout handlers,” on page 199)
- Deploy a Hub configuration (Chapter 2, “Managing projects and Hub connections,” on page 11)
- Generate Flex components (refer to the IBM Initiate Master Data Service Composer Guide for JavaScript Development)
- Create a job set (Chapter 3, “Jobs and job sets,” on page 21)
- Validate local weights (Chapter 6, “Weight generation,” on page 175)
- Rescore members in sample pairs file(s)
- Upgrade Initiate project artifacts from previous Workbench versions (Chapter 2, “Managing projects and Hub connections,” on page 11)
- Register Hubs to projects (Chapter 2, “Managing projects and Hub connections,” on page 11)

#### Rescoring members in sample pair files

**About this task**

If you change the algorithm after running Generate Threshold Analysis Pairs ("Database Tools jobs” on page 78), you can rescore the sample pairs without having to rerun the job.
**Procedure**

1. From the menu, select **Initiate -> Rescore members**. The Rescore Members dialog opens.
2. Select the Project, Hub, Member Type and Entity Type as needed, then click **Next**.
3. Click **Browse** beside the Sample Pair Files box to navigate to the sample pair file(s) previously generated.
4. Click **Finish** to begin rescoring, or click **Next** to view the summary page prior to clicking **Finish**. The sample pair files are updated with the new scores.

---

**First step in using IBM Initiate Workbench**

The first step in performing Hub configuration tasks is to create an Initiate project. Refer to Chapter 2, "Managing projects and Hub connections," on page 11 for instructions on creating a project and importing a configuration from the Hub.
Chapter 2. Managing projects and Hub connections

About this task

In IBM Initiate Workbench, a project is merely a container that holds the Hub configuration and its associated files. Before you import a Hub configuration into a project, you must first create a new project or import an existing project.

The following sections describe how to use Workbench to:

- Create a Hub connection (see “Adding a connection”)
- Create a new project (see “Creating a new project” on page 13) or Import an existing project (see “Importing an existing project” on page 14)
- Import a Hub configuration into an existing project (see “Importing a Hub configuration” on page 15)
- Deploy a Hub configuration (see “Deploying a Hub configuration” on page 17)
- Upgrade the artifacts from an existing Initiate project to a new Workbench version (see “Upgrading Initiate projects” on page 18)

Hub connections

About this task

A connection to the Master Data Engine instance is required in order to download and upload IBM Initiate Workbench project files. These topics describe various connection management steps.

Adding a connection

About this task

To add a connection to the Master Data Engine instance, follow these steps.

Procedure

1. From the menu, select Initiate > Register Hubs to Projects.... The Register Hubs to Projects dialog opens.
2. If you have already created a project, select the Project from the drop-down list to which you wish to associate the Hub.
3. The section labeled Available Hubs lists the Hubs that have been registered, if any. To add a Hub to this list, click Edit.... The Hub Connections dialog opens.
4. Click Add.... The Add Hub Connection dialog opens.
5. On the Add Hub Connection dialog, set the following options:
   - **Connection Name**: Type a description for the connection (e.g., Production or Test).
   - **Host**: Type the machine name or IP address for the server where Hub software is running.
   - **Port**: Type the port number for the Hub.
   - If your Master Data Engine instance is configured to provide MPINet communications over HTTP, select the Use MPINet over HTTP check box.
• If you are using SSL security, select the **Use SSL for MPINET** check box and select the **SSL Version** from the list. If you want Workbench to verify the SSL certificate returned from the Hub, click **Verify SSL Certificate**.

• **JMX Port:** Type the port number to use for JMX monitoring. If desired, enable the **Use SSL for JMX** option.

6. Click **OK**. Repeat steps 4-6 as needed.

7. Click **Close** on the Hub Connections dialog.

8. On the Register Hubs to Projects dialog, check the server(s) to which you wish to associate the selected project, and then click **Finish** or **Cancel**.

**Results**

The next step is typically **“Creating a new project” on page 13.**

**Editing a connection**

**About this task**

To edit a connection to the Master Data Engine instance, follow these steps.

**Procedure**

1. From the Initiate menu, click **Register Hubs to Projects**. The Register Hubs to Projects dialog opens.
2. The section labeled Available Hubs lists the Hubs that have been registered. To edit a server in this list, click **Edit**. The Hub Connections dialog opens.
3. On the Hub Connections dialog, highlight the connection you want to update, and then click **Edit...**
4. Make the necessary changes to the Connection Name, Host Port, MPINet over HTTP, SSL and JMX options. Click **OK**.
5. Click **Close** on the Hub Connections dialog.
6. On the Register Hubs to Projects dialog, click **Finish**.

**Removing a connection**

**About this task**

To remove a connection to the Master Data Engine instance, follow these steps.

**Procedure**

1. From the menu, click **Initiate > Register Hubs to Projects**. The Register Hubs to Projects dialog opens.
2. The section labeled Available Hubs lists the Hubs that have been registered. To delete a Hub from this list, click **Edit**. The Hub Connections dialog opens.
3. On the Hub Connections dialog, highlight the connection you want to delete, and then click **Remove**.
4. Click **Yes** to remove the selected connection.
5. Click **Close** on the Hub Connections dialog.
6. On the Register Hubs to Projects dialog, click **Finish**.
Managing projects

About this task

The data that comprises a hub configuration is managed within IBM Initiate Workbench projects. Before you can create or edit a hub configuration, you need one or more projects.

Creating a new project

About this task

To create a IBM Initiate Workbench project, follow these steps.

Procedure

1. From the Initiate menu, select New Initiate Project...
2. In the Project name field, enter a name for your new project. If the Use default location option is checked, the project will be created in the current workspace directory. If not, specify a location outside of the current workspace (such as another local drive or network drive) using the Browse button. Click Next.
3. On the Registered Hubs page, select the Hub(s) to which you wish to associate the project. If the desired Hub is not listed, refer to “Adding a connection” on page 11. Click Next.
4. On the Templates page, select a template to use. Available templates are:
   - **Default**
     This template contains predefined attribute type definitions, link types, task types, etc. It does NOT contain any member type, entity type, or source definitions.
   - **Empty**
     Using this template creates an empty project. It contains no member type, entity type, source definitions, predefined attribute type definitions, link types, task types, etc. To populate this project, you must import a configuration from an existing Master Data Engine instance.
   - **Initiate Demo Person**
     Using this template creates a demo project that has one member type called “PERSON.” It also includes a sample algorithm for the “PERSON” member type.
   - **Initiate Demo Householding**
     Using this template creates a demo project that has one member type called “PERSON” and two entity types, id (identity) and hh (householding). It includes:
     - a sample algorithm for the “PERSON” member type that demonstrates the recommended comparison and bucketing functions and a sample set of weights that you can reference in implementing a Household Solution.
     - a Householding job set template containing recommended job sequences and options that can be used.
     - a Basic Householding Configuration cheat sheet that highlights the artifacts in this template project and aids in understanding how to implement the Household solution (see “Running an Initiate-supplied cheat sheet” on page 352).
     - a sample Clover.ETL graph for performing a Household data extract.
5. Click Finish.
Results

IBM Initiate Workbench then creates the project and adds the following directories under the workspace directory:

- **.settings**
  - contains files used by IBM Initiate Workbench to store preference settings

- **anonutil**
  - contains sample default value files and filter files used by the Anonymous Value Utility

- **handlers**
  - contains scripting support for packaging Java handlers

- **lib**
  - contain any additional Java code library files needed for deployment (.jar)

- **src**
  - contain any additional Java source files needed (.java)

Additional directories are added if you select the default or demo templates, or import a hub configuration after creating an empty project.

- **client**
  - contains a generic api.properties file for use with the ESOA Toolkit. Refer to the IBM Initiate Enterprise Service Oriented Architecture (SOA) Toolkit User’s Guide for more information.

- **inspector**
  - contains IBM Initiate Inspector configuration-related files.

- **strings**
  - string value files associated with string codes defined in the hub configuration

The next step is typically "Importing a Hub configuration" on page 15.

Importing an existing project

About this task

To import an existing project, either switch your workspace to the project's parent directory using File > Switch Workspace... or copy the project files into your current workspace.

To copy the project files into your current workspace:

Procedure

1. Using a file browser, copy the existing project directories into the workspace directory on your file system.
2. From the Navigator view, right click the project and select Import....
3. On the Import:Select dialog, select Existing Projects into Workspace. Click Next.
4. On the Import:Import Projects dialog, Select root directory is the selected default. Use the Browse button to specify the location of the previously exported project(s). If desired, you can also select to import from an archive file (.jar, .zip, .tar, .tar.gz, .tgz) by choosing the Select archive file option instead.
5. Click the checkbox for each project you wish to import.
6. If you wish to copy the projects into the current workspace, check the Copy projects into workspace option. This option places a copy of the imported project's folders and files into the current workspace directory.

7. Click Finish to import the project. It now appears in the Navigator view.

Renaming a project
About this task

To rename a project, right-click the project name in the Navigator view and select Rename. Type the name of the project as desired. When you press [Enter], the new project name is accepted. If the project has a corresponding Hub configuration file (.imm), it is renamed to match the new project name.

Deleting a project
About this task

If you no longer need a project, you can delete it from IBM Initiate Workbench.

Procedure
1. Right-click the project in the Navigator view. The context menu opens.
2. Select Delete.
3. A dialog opens asking if you want to delete all files in the project folder, or just remove the project from IBM Initiate Workbench without deleting the files. Select the desired options, and then click Yes.
4. If the project has previously-executed jobs associated with it, indicate whether you want to delete any job-related artifacts residing on the Hub.

Managing Hub configurations
About this task

Once a project is created, you need to import a Hub configuration from the Master Data Engine.

Importing a Hub configuration
Procedure
1. From the menu, click Initiate > Import Hub Configuration.... The Import Hub Configuration wizard opens.
2. Select the Project into which you wish to import the configuration information. The project currently selected in the Navigator view is the default.
3. Select the Hub you wish to import from. To add or edit the Hubs in this list, consult "Adding a connection" on page 11.
4. Select the following configuration artifacts as desired.

Note: Some artifacts have dependencies on other artifacts and must be selected together.

Hub core configuration retrieves the basic data about the Hub and dictionary data, such as member types and attribute types.

All Hub algorithms retrieves all algorithms configured on the Hub.
Specific algorithms defined by your implementation can also be imported individually. If the Hub core configuration has previously been imported, any algorithms included in the dictionary data will be listed in the group box within the Algorithms section.

**String data files**
retrieves string code descriptions as well as text files that define the string codes. When selected for import into a new project, Hub core configuration and Hub metadata must also be selected.

**Weight tables**
retrieves contents of the weight tables in the Hub database, if any. This option facilitates the transfer of a more complete configuration from one environment to another, such as development to test.

**Inspector configuration**
imports only the IBM Initiate Inspector configuration files.

**DataTrust configuration**
imports only the IBM Initiate Master Data Service Data Trust configuration files.

5. If you want to automatically open the .imm (configuration) file when the import is finished, enable the **Open Hub configuration after import** option.

6. Click **Finish**. The Login dialog opens.

7. Enter your user name and password to retrieve the Hub configuration information from the Master Data Engine. Click **Log In**. The Hub metadata is imported along with any additional data, such as algorithms, as selected.

**Results**
The user name used to log in to import and deploy a configuration must have administrative privileges within the LDAP system. Refer to [Chapter 9, “User management,” on page 207](#) for more information.

**Opening a Hub configuration**

**About this task**
To manipulate the data that comprises a Hub configuration, you must first open the project's .IMM file in the IBM Initiate Workbench editor.

**Procedure**
Double-click the *project_name*.imm file in the Navigator view.

**Results**
The Initiate Configuration (*project_name*) opens in the editor.

**Note:** If the Navigator view does not list a *project_name*.imm object, re-import the Hub configuration and make sure the **Import the Hub metadata only** checkbox is unchecked.

The categories shown in the left pane contain the available components:

- [“Applications” on page 126](#)
- [“Attribute types” on page 107](#)
Saving a Hub configuration

About this task

After making desired changes to a Hub configuration, it must be saved to the local
file system. In order for a configuration to be uploaded to a Master Data Engine
for use in a test or production environment, its IBM Initiate Workbench project is
“deployed” to that Hub. Refer to “Deploying a Hub configuration” for detailed
information on deploying projects.

Procedure

To save a configuration, perform one of the following actions:

- Click the Save icon on the IBM Initiate Workbench toolbar.
- Press [Ctrl][S] on the keyboard.
- Select File > Save from the main menu.

Results

Hub configurations are stored locally by IBM Initiate Workbench in the workspace
directory in a child directory under the project’s name. The workspace directory is
created when IBM Initiate Workbench is started. Configuration file names must
match their project names:

```
..\workspace\ project_name\project_name.imm
```

Should you wish to rename a project, use the Rename action, which updates the
Hub configuration file to match the new project name. Refer to “Renaming a
project” on page 15.

Deploying a Hub configuration

About this task

Once the configuration objects contain the desired data, you can upload the
configuration file to a Hub using the Deploy option. Before deploying, ensure that
no errors are listed in the Problems view.
Procedure

1. To deploy the configuration to a Hub registered to the project, select **Initiate > Deploy Hub Configuration** from the main menu. The Deploy Hub Configuration wizard opens.
2. Select the project to be deployed. The current project is the default.
3. Select the Hub to which you wish to deploy the project.
4. Select the following configuration artifacts as desired. All options are enabled by default.
   - **Core configuration** - deploys the dictionary data, such as member types and attribute types.
   - **Algorithm(s)** - deploys all algorithms configured on the Hub. This is automatically deployed when relationship rules exist.
   - **Inspector configuration** - deploys the IBM Initiate Inspector configuration files.
   - **String data** - deploys string code descriptions and the text files that define the string codes.
   - **Weight data** - deploys contents of the local weight table files. This option is used to insert weight data (located in the directory shown in **Weights folder** into the Hub database after generating weights.
   - **Flexible Search configuration** - deploys the IBM Initiate Flexible Search configuration files.
5. On the Options tab, indicate whether you wish to **Generate DDL instead of creating/dropping tables**. The default is disabled.
6. Click **Finish**. When the deployment begins, the saved configuration is verified against the Hub. If validation errors are found by the Master Data Engine, a warning dialog opens requesting that you fix the errors prior to deployment. A progress dialog displays the current deployment progress.

Results

Hub configurations can also be deployed using a job. Refer to [Deploy Hub Configuration](#) on page 83.

Upgrading Initiate projects

About this task

When you have existing Initiate projects created using an earlier version of IBM Initiate Workbench, you may see a message stating that the project is out of date when you open an artifact (hub configuration, algorithm, etc) in that project. Follow the procedure below to upgrade the project.

Procedure

1. From the menu, select **Initiate > Upgrade Initiate Project**. The Upgrade Project wizard opens.
2. Select the project to be upgraded. The current project is selected by default. Click **Next**.
3. The outdated project artifacts are identified by “Yes” in the **Upgrade Needed** column. Select an artifact that needs updating, and then click the **Upgrade** button on the right.
**Note:** Some artifacts have dependencies on other artifacts, so it is important to upgrade artifacts (files) as they are ordered top to bottom.

For example, the .classpath file is normally fixed automatically as part of the engine-metadata.xml upgrade step. But if you upgrade the IBM Initiate Workbench project .IMM file prior to running the engine-metadata.xml upgrade step, you would need to manually edit the .classpath file in the root of the project(s) to remove the two classpath entry lines for "fis-server.jar" and "commons-jelly-1.0.jar."

4. The Element Upgrade wizard opens, describing the first step that needs to be performed. Click **Perform Upgrade Step**.

5. When all steps have been performed, click **Finish**. You return to the Upgrade Project Content page.

6. Repeat steps 3-5 for each project artifact that needs to be upgraded. When the **Upgrade Needed** column shows “No” for all artifacts, click **Finish**.

**Results**

You can now open the desired artifacts in IBM Initiate Workbench. The next step in the implementation process is typically configuring the member model (see [Chapter 4, “Configuration editor,” on page 89](#)).
Chapter 3. Jobs and job sets

A job is a task or utility performed by the Hub. Job sets are groupings of one or more jobs. For jobs that are run frequently, you can set up job templates to use later. Tasks that can be run via the Jobs view include:

- Deploying a configuration to the Hub
- Executing utilities such as mpxcomp, mpxprep, and mpmlink
- Performing member comparisons
- Creating threshold analysis pair spreadsheets
- Retrieving a file from the Hub

Job results are stored on the Hub server by project. In many cases, job results can be retrieved or viewed from the server by right-clicking the completed job and selecting Get Job Results. Other IBM Initiate Workbench instances with a project whose name matches the project under which jobs were run can retrieve or view job results in this manner, provided the matching projects are configured for the same Hub.

Storage files

When jobs such as Load UNLs to DB (madunlload) or Derive Data and Create UNLs (mpxdata) are run, database tables are typically created in the default users workspace. As a result, the users data file may get full. You can specify a tablespace and indexspace to be used instead by directing the Hub to use a storage file. When utilities are executed from the command line, this storage file (stofile) name and location are read from an environment variable. Jobs (utilities) that are run through IBM Initiate Workbench read the storage file location from the com.initiate.server.system.cfg file instead. Add the MAD_STOFILE property to the com.initiate.server.system.cfg file and set it to match the MAD_STOFILE environment variable. For example:

MAD_STOFILE=${mad.root.dir}/sql/filename.sto

or

MAD_STOFILE=C:/temp/filename.sto

For more information on the use of storage files, consult the IBM Initiate Master Data Service Engine Installation Guide. Consult the database documentation for details on stofile syntax.

Data derivation

Derived data is information that has been “configured” for matching and scoring. When data is derived it goes through standardization and bucketing. The derived version of the data is stored in addition to the original version. Derived data will not include attribute tokens which were identified as “Anonymous Values.” Likewise, common conversions are made to data to ensure consistent formatting (e.g., special characters are removed from identifiers, such as hyphens (-) in SSN values, and the area code is removed from United States telephone numbers, leaving a 7-digit number).

There are two types of derived data:
• **Comparison Strings**: These are carat (^)-delimited strings with standardized data for each field or attribute referenced in the algorithm. Each member record correlates to a single comparison string.

• **Bucket Hashes**: These are numbers that represent the buckets to which each member belongs. Each member can have multiple hash assignments.

The data derivation process includes four main events:

• raw data is parsed into segment specific unload files (optional, depending on derivation method)
• comparison strings are built from standardized data
• members are assigned bucket hashes
• binary files are created for faster computation (optional, depending on derivation method)

There are two main ways the data derivation process is triggered: automatically on insert/update of a member record, and manually running an engine utility in the Jobs view (or on the command line). The manual derivation process is most commonly performed when running an initial load of the database, while generating weights, or when you want to propagate a change to your algorithm data that already resides in the Hub’s database. The methods to derive data are:

• **Derive Data and Create UNLs (mpxdata)** is typically used the first time you derive data for the initial load. It takes raw data and builds member unload files (using a .cfg file that you create), generates comparison strings, assigns bucket hashes, and creates binary files for faster comparison. The unload (.UNL) files can then be loaded into the Hub using Load UNLs to DB (madunlload).

• **Prepare Binary Files (mpxprep)** is usually used with an Incremental Cross Match (IXM). In this process, you load the records from each of the sources separately, and then run Prepare Binary Files (mpxprep) to compile the binary files. If you compiled binary files for each source separately, they would not be complete. Running Prepare Binary Files (mpxprep) as a final step reads all of the records from the member tables and builds a complete set of bulk cross match (BXM) files.

• **Derive Data from UNLs (mpxfsdvd)** is typically used during the implementation process (before the Hub goes live) when you have made changes to the algorithm, but have not received new records since the last time you ran Derive Data and Create UNLs (mpxdata). It uses pre-existing member unload files to extract and create comparison strings, makes bucket hash assignments, and creates binary files.

• **Derive Data from Hub (mpxredvd)** is typically used after the Hub has gone live. If you change part of the bucketing or comparison strategy in the algorithm, then you will need to re-assign your bucket hashes and comparison strings. The Derive Data from Hub (mpxredvd) method reads the data in the database tables, since they include all of the updates up to that point in time. In this case, if you have received updates (through brokers or the API), your old UNL files would not reflect those updates, but the tables would.

• **Member Model Transform Graph** is a Clover.ETL-enabled wizard that guides you through the process of creating member UNL files from your data extract. Once done, you have a Clover.ETL graph that you can run to parse out the member UNL files. The benefit is that you can use existing metadata from the extract cleansing process instead of designing a .cfg file. You would follow this step with Derive Data from UNLs (mpxfsdvd) to create bucket hashes, comparison strings, and binary files.
After deriving data, you need to load the results into the Hub database (except in the case of Derive Data from Hub (mpxredvd), which updates the database directly). This is done using Load UNLs to DB (madunlload).

**Bulk cross match**

The bulk cross match (BXM) is an off-line process that allows you to compare and link thousands of records per second. The typical Hub implementation includes a BXM process to establish entities based on the starting source system records. New and updated source system records are processed through entity management to maintain entities going forward. If you are not using entities in your implementation, you would not need to perform a BXM.

The BXM is most commonly performed in the initial stage of the implementation and again just before the system goes live. It compares your member records in binary form and then measures the comparison scores against your thresholds to create entity assignments and tasks. Matching and linking in bulk means that the Hub will need to focus on matching and linking only the new records that come into the database after the Hub is in production.

The following must be in place in order to perform the bulk cross-match:
- derived data
- weights
- Hub engine
- algorithm
- data dictionary

The BXM process uses the weights to generate an aggregate comparison score, which is compared to the thresholds to determine auto-linking and task generation. Two primary jobs make up the BXM process: mpxcomp and mpxlink. After running the compare and link, the data will need to be loaded into the database.

**Creating a Job Set**

**About this task**

There are three ways to create a new job set in IBM Initiate Workbench:
- From the Initiate menu, select **New Job Set**.
- On the toolbar, click **Create a new job set**.
- In the Jobs view, click **Create a new job set**.

The New Job Set wizard opens.

**Procedure**

1. Select the Project for which you wish to create a job. The default is the current project.
2. Select the **Job Set Template** you wish to run, if any. If this is a new job set, leave this set to None. Job set templates are stored in the project's jobTemplates folder in XML format for ease of reuse.
3. Select the Hub on which you wish to run this job. Only Hubs that have been associated with the selected project are listed. If you need to add or edit a Hub, click **Edit**. Refer to Chapter 2, "Managing projects and Hub connections," on page 11 for details.
4. Enter the **Work Directory**, if any, for this job. This is where files related to the current project are stored and accessed on the computer where the Engine is running. The Work Directory can be used to provide a second level of separation or partitioning of the job results associated with a particular project. By default, this path is relative to the Hub instance directory’s ‘work’ directory (`hub_instance_path\inst\mpinet_hub_instance_name\work\`). If specified, the value in this field is appended to the project name with a tilde character as a delimiter:

```
hub_instance_path\inst\mpinet_hub_instance_name\work\projectname~workdir
```

If the Work Directory is not specified on the job wizard, job files related to the current project are stored in `hub_instance_path\inst\mpinet_hub_instance_name\work\projectname`.

5. If you wish to save this job set as a template for future use, enable the **Save job set as a template** option, then supply a **Template Name** and **Template Description** in the fields provided.

6. Click **Next**.


8. Click the + icons to reveal the jobs available for each job category. When you select a job, its description appears in the right pane. Choose the job you wish to add to this job set and click **OK**.

9. The Job Configuration screen displays the job(s) you selected, along with its options. Refer to the sections below for details on configuring the options for each job. Repeat steps 7-9 as needed to add more jobs to this job set.

10. Click **Move Up** or **Move Down** to sort the jobs in the order in which they are to be executed. Click **Remove Job** to remove unwanted jobs from the job set.

**Note:** The results of some jobs are used as input for other jobs and must be retrieved from the server using the Get Job Results action prior to running the subsequent job, and so not all jobs can be chained.

11. Click **Finish** to execute the job set or **Save** to save the template without executing the job set. The **Save** button is only enabled if you selected the **Save job set as a template** option above and the job set had changes.

12. If you selected **Finish**, you are asked to log in to the specified Hub. Provide the user name and password, and click **Log In**. The job is then uploaded to the Hub to be executed.

13. Once the job has finished running, the **Status** column in the Jobs view shows **SUCCESS** or **FAILED**. You can drill into the job set to view the job results.

14. If you need to retrieve job results from the hub, right-click the desired job within the job set and select “Get job results.”

15. If you want to rerun a job set, right-click the job set and select “Run job set again.” The job set wizard opens with the selected job set’s jobs and options pre-selected. Simply click **Next** and Finish to launch the job set again.

**Job set execution**

When jobs are saved into a job set, you can execute the entire job set easily by selecting it from the Job Set Templates section in the job wizard. All jobs in the job set will be executed.

In addition, job sets can be run using the `madconfig run_jobset` command. If IBM Initiate Workbench is running on a different computer than the Hub engine, you will first need to copy the job set’s XML file from the IBM Initiate Workbench...
project to the Hub server. It is located at workspace\projectname\jobTemplates. If the job set includes the Deploy Hub Configuration job, the contents of the entire project will also need to be copied to the server on which the Hub is running. Refer to the “Using Utilities” chapter of the IBM Initiate Master Data Service Engine Installation Guide for information on using this command.

**Note:** The Generate Threshold Analysis Pairs and Get File jobs cannot be run using the madconfig run_jobset command. If an unsupported job is present in the job set, the madconfig utility will skip execution of that job and log an error in the jobsetrunner- date - id .mlg file.

**Job files and custom Work directories**

The Work directory is where job files related to the current project are stored and accessed on the computer where the Master Data Engine is running. The default Work directory is:

```
hub_instance_path\inst\mpinet_hub_instance_name\work\n```

such as:

```
C:\myhubinstance\inst\mpinet_myhubinstance\work\n```

When a job is executed from IBM Initiate Workbench, it always adds the `projectname` directory and, if you specify a "Work directory" in the New Job Set wizard, the ~workdir portion. Both the `projectname` and ~workdir values are passed to the Hub in the job set information submitted by IBM Initiate Workbench.

---

**Job Categories**

Most utilities can also be executed on the command line. The options for the IBM Initiate Workbench job and command line utility are both shown in the tables that follow. Comprehensive usage information is also accessible from the command line by running the utility without any other strings (for example: utility.exe or utility.bat, where utility is the utility's name).

Most command line utilities are run from the engine_install\bin directory, but a few of them are run as MADCONFIG “targets” from the hub_install\scripts directory. In addition, job sets can be run using the MADCONFIG run_jobset command. Refer to the “Using Utilities” chapter of the IBM Initiate Master Data Service Engine Installation Guide for more information.

**Job Categories**

<table>
<thead>
<tr>
<th>Job Categories</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Algorithm &amp; Analysis jobs&quot;</td>
<td>39</td>
</tr>
<tr>
<td>&quot;Bulk Tools jobs&quot; on page 39</td>
<td></td>
</tr>
<tr>
<td>&quot;Database Tools jobs&quot; on page 78</td>
<td></td>
</tr>
<tr>
<td>&quot;Hub Administration jobs&quot; on page 83</td>
<td></td>
</tr>
<tr>
<td>&quot;Job Set Templates&quot; on page 87</td>
<td></td>
</tr>
</tbody>
</table>

**Algorithm & Analysis jobs**

The following jobs are grouped under Algorithm & Analysis. Most of the algorithm and analysis jobs have a tab on which you can specify the logging options, which are the same for all jobs.

- Compare members
- Generate Frequency
- Generate Threshold Analysis Pairs
- Generate weights
- Prepare Bucket Analysis
- Validate Weights

**Compare Members (mpimcomp)**

Executes the MEMCOMPARE interaction to retrieve a text representation of Master Data Engine results of comparing two members. Direct comparisons of attributes for the first member (proband) are compared against those of the second member (candidate) to determine if the two records represent the same person.

*Table 1. Compare Members (mpimcomp) options*

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type</td>
<td>entityType</td>
<td>Compare Members (mpimcomp) can be run for one entity type at a time, selected in the Entity Type field. All member types within the specified enttype are processed.</td>
</tr>
<tr>
<td>First Member</td>
<td>memRecno1</td>
<td>The member to be compared against. This is a MEMRECNO of a member of the type shown in the Entity Type field.</td>
</tr>
<tr>
<td>Second Member</td>
<td>memRecno2</td>
<td>The member to compare to the First Member. This is a MEMRECNO of a member of the type shown in the Entity Type field.</td>
</tr>
</tbody>
</table>

**Note:** The entityType, memRecno1 and memRecno2 options are positional; when used on the command line, their option names are not specified. An example of this usage is:

```
mpimcomp id 155 247
```

When the job has finished, right-click the completed job in the Jobs view and select **Get Job Results**. The text comparison opens.

The value listed for “mcc” indicates the match type. Possible match types, based on the attribute comparison score, are:
- E = equal (the values are the same)
- D = different (the values are different)
- P = partial (values are partially the same)
- M = missing (value is missing)
- X = EQWRD (equal at a ‘word’ or token level)
- Y = EQINI (Equal at a single character initial level)
- P = phonetic (values have a phonetic match)
- N = nickname (nickname value)
- O = NICKMETA
- I = initial (Equal at an initial level, but not a direct match of a single character.
- A = acronym
- C = compound match
- 1 = Prefix match with prefix in 1st member
- 2 = Prefix match with prefix in 2nd member
You may also see a value of \( T \) (true) or \( F \) (false). This occurs if the False Positive Filter (FPF) feature is used:

- \( T \) indicates that the FPF triggered, and the algorithm believes the records are not the same person.
- \( F \) indicates that the FPF was not triggered, and the records may be the same person.

The associated score is dependent on the settings in the weight table.

**Generate Frequency Stats (mpxfreq)**

Generates data from which attribute frequencies can be derived. This job requires derived data as input, such as that generated by mpxdata, mpxprep, mpxfsdvd, or mpxredvd.

The Generate Frequency Stats (mpxfreq) job has three purposes:

- The *Generate raw frequency table output* option is for counting the values that are checked to determine whether they are anonymous.
- The *Generate frequency tables for frequency-based bucketing* option is to enable the Frequency-Based Bucketing (FBB) feature in which you count the number of members in each bucket, and then determine the buckets that exceed the Maximum Bucket Frequency number (set in the Bucket Group Properties in the Algorithm).
- The third use for Generate Frequency Stats (mpxfreq) is to *Generate frequency tables for weight generation*.

**Table 2. Generate Frequency Stats (mpxfreq) options**

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type</td>
<td>-entType</td>
<td>If you have multiple entity types in the Hub database and only need to compute frequencies for one of those entity types, the Entity Type filter may be used. All member types within the specified enttype are processed.</td>
</tr>
<tr>
<td>Member Type</td>
<td>-memType</td>
<td>If you have multiple member types in the Hub database and only need to compute frequencies for one of those member types, the Member Type filter may be used. All entity types for that member are processed.</td>
</tr>
<tr>
<td>Generate raw frequency table output</td>
<td>-rawMode</td>
<td>Instructs Generate Frequency Stats (mpxfreq) to count the values that are checked to determine whether they are anonymous. (This option was previously known as anonMode.) It provides the numerical input for the “Anonymous Value Utility” on page 144. You must have anonymous string codes configured, and they must be referenced in the Standardization functions in your Algorithm before you can run the mpxfreq utility to generate output to feed to the Anonymous Values Utility. <strong>Note:</strong> This option does not generate the files needed for weight generation; when running the Generate Weights job, start from the “Delete artifacts from previous run” option.</td>
</tr>
<tr>
<td>Workbench</td>
<td>Command line</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Generate SQL output</td>
<td>-cmpFreq Sql</td>
<td>Instructs Generate Frequency Stats (mpxfreq) to generate SQL files in the \frq subfolder. Users typically do not need both SQL output and UNL output.</td>
</tr>
<tr>
<td>Generate UNL output</td>
<td>-cmpFreq Unl</td>
<td>Instructs Generate Frequency Stats (mpxfreq) to generate UNL files in the \frq subfolder. Users typically do not need both SQL output and UNL output.</td>
</tr>
<tr>
<td>Generate frequency tables for frequency-based bucketing</td>
<td>-fbbMode</td>
<td>Uses the derived bucket data to generate a list of string frequencies. If a given string is over the maximum limit, it is added to a list in the dictionary and will not be used for candidate selection. Limits are specified in the algorithm Each bucket role can have a different limit Static process -- you must re-run this step as your data set changes and grows. If any strings are added to strFreq, then you should re-derive the bucket data in order to reduce the number of comparisons required by mpxcomp.</td>
</tr>
<tr>
<td>Generate SQL output</td>
<td>-bktFreqSql</td>
<td>Instructs Generate Frequency Stats (mpxfreq) to generate SQL files in the FRQ output directory. Users typically do not need both SQL output and UNL output.</td>
</tr>
<tr>
<td>Generate UNL output</td>
<td>-bktFreqUnl</td>
<td>Instructs Generate Frequency Stats (mpxfreq) to generate UNL files in the FRQ output directory. Users typically do not need both SQL output and UNL output.</td>
</tr>
<tr>
<td>Generate frequency tables for weight generation</td>
<td>-wgtMode</td>
<td>This option is identical to running the “Generate counts for all attribute values” option in the Generate Weights job with “Execute all remaining steps through end of process” disabled. The advantage to using this option rather than simply executing the Generate Weights script functionality is to enable you to take advantage of the Performance Tuning options when generating the frequency tables. When using this option, be sure to direct the FRQ output to the correct directory; the Generate Weights job typically expects to find it in the weights\frq subdirectory. If you later run Generate Weights starting with “Generate random pairs of members,” you will need to first create the necessary directories for the weight generation output.</td>
</tr>
<tr>
<td>BXM input directory</td>
<td>-bxmInpDir</td>
<td>The bxm folder that contains the comparison binary work files. These files were produced by one of the data derivation methods discussed in “Data derivation” on page 38, so this path should match the BMX output directory specified by the selected derivation process.</td>
</tr>
<tr>
<td>FRQ output directory</td>
<td>-frqOutDir</td>
<td>Specifies the output directory to which frequency results are written.</td>
</tr>
<tr>
<td>Encoding</td>
<td>-encoding</td>
<td>Choices are Latin1, UTF8 and UTF16. Default: latin1</td>
</tr>
<tr>
<td>Workbench</td>
<td>Command line</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Number of threads</td>
<td>-nthreads</td>
<td>This value should correspond to the number of CPUs available on the Hub. The goal is to take advantage of all the processing resources available. For example, if running the Hub on a computer with four CPUs, set the number of threads to 4 to keep all four CPUs busy and minimize the time Generate Frequency Stats (mpxfreq) takes to run. If you were to set it to 2, only two CPUs would be used, and the processing time would be longer. Setting it too high would cause the Hub to switch back and forth between running threads and threads waiting for available cpu cycles.</td>
</tr>
<tr>
<td>Number of member partitions</td>
<td>-nMemParts</td>
<td>This should correspond to the number of member partitions used when running mpxdata, mpxprep, mpxredvd, or mpxfsdvd (these utilities can also produce the starting bxm data).</td>
</tr>
<tr>
<td>Primary hash slot count</td>
<td>-hash1Cnt</td>
<td>This is the number of slots available in the hash table. The larger the diversity of the dataset, the greater the advantage in increasing this number. Each slot maintains a list of string values with their counts. The fewer string values that need to be parsed in a slot, the faster the program will run. Increasing this number also increases memory requirements.</td>
</tr>
<tr>
<td>Compression hash slot count</td>
<td>-hash2Cnt</td>
<td>This is the number of slots available during the join of multiple threads into the master frequency list. All records have been processed and the Hub is simply adding up the totals. This number will typically be smaller than Primary hash slot count. Like Primary hash slot count, increasing this number also increases memory requirements.</td>
</tr>
<tr>
<td>Slot compression ratio</td>
<td>-h12CvPct</td>
<td>This option allows the hash tables to be compressed down if the ratio of total nodes to nodes per nway falls above the percentage specified.</td>
</tr>
<tr>
<td>Page size</td>
<td>-pageSize</td>
<td>This is the amount of memory in megabytes initially allocated for string nodes. This value should be based on the size of the strings being frequenced and the number of unique string values.</td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Common audit record number for all UNL/SQL output</td>
<td></td>
<td>Indicates the value to use for the AUDRECNO column in the SQL or UNL file.</td>
</tr>
<tr>
<td>Filter by Source</td>
<td>-refSrcCode</td>
<td>Enables frequency analysis to be performed on a single source.</td>
</tr>
<tr>
<td>Log Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trace logging</td>
<td></td>
<td>Produces a trace of activity as interactions flow through the system. This option is very verbose and should only be used for short periods of time.</td>
</tr>
</tbody>
</table>
Table 2. Generate Frequency Stats (mpxfreq) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug logging</td>
<td></td>
<td>Produces low-level diagnostics used internally by Initiate Systems to identify what was happening on the system before an error condition occurred. This option generates a large amount of output per activity and should only be used for short periods of time. <em>Note:</em> Debug logging can potentially include personal member information such as member identification number, name, etc.</td>
</tr>
<tr>
<td>Timer logging</td>
<td></td>
<td>Produces timings on certain operations to help identify where significant processing time is elapsing.</td>
</tr>
<tr>
<td>SQL logging</td>
<td></td>
<td>Outputs the SQL that is sent by the Master Data Engine database layer to the RDBMS. This helps in diagnosing database-related issues. This option can produce large amounts of output depending on the activity.</td>
</tr>
<tr>
<td>Audit logging</td>
<td></td>
<td>Produces activity information and non-critical warnings. Often, this option is used when a new system is first implemented to monitor activity.</td>
</tr>
<tr>
<td>Algorithm logging</td>
<td></td>
<td>A separate logging level for algorithm-related debug information without the risk of including protected health information (PHI). Default: disabled</td>
</tr>
</tbody>
</table>

**Generate Threshold Analysis Pairs**

Run the Compare Members in Bulk (mpxcomp) and Link Entities (mpxlink) utilities prior to running Generate Threshold Analysis Pairs.

This job is a cross-match program to generate random pairs for weight generation. It creates one or more sample pairs files in XLS format, which can be read by the Algorithm Editor’s Threshold Calculator to perform “statistical FPR based on evaluated sample pair data.” This is executed on the command line using mpxpair.

*Note:* Sample pairs should be created on a sample of data up to a few million rows and will likely not work on larger datasets. Sample pair results over a few million rows would be statistically identical to a smaller subset of data. There is no reason to run sample pairs on the full subset of data.

If the algorithm is changed after running Generate Threshold Analysis Pairs, you can rescore the sample pairs by using the **Initiate > Rescore members in sample pairs file(s)** menu. Refer to “Rescoring members in sample pair files” on page 9.

*Note:* The Generate Threshold Analysis Pairs job cannot be executed in a job set using the madconfig run_jobset command.
Table 3. Generate Threshold Analysis Pairs options

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type</td>
<td>-entType</td>
<td>Sample pairs can be generated for one entity type at a time by specifying it in the Entity Type field. All member types within the specified enttype are processed.</td>
</tr>
<tr>
<td>Input Directory</td>
<td>-bxmInpDir</td>
<td>Where .bin input files will be located. This directory is relative to the instance Work Directory on the server hosting the Hub. Default: bxm</td>
</tr>
<tr>
<td>Number of pairs per score</td>
<td>-npairs</td>
<td>Instructs the utility to generate this number of pairs per score in the selected score range. If the dataset contains an inadequate number of pairs, the value selected here will not be returned in the job results. Default: 10 Maximum: 1000</td>
</tr>
<tr>
<td>Include only cross source pairs?</td>
<td></td>
<td>When selected, this option forces the job to create sample pairs from different sources, rather than from the same source.</td>
</tr>
<tr>
<td>Minimum Score</td>
<td></td>
<td>The Score Range specifies the minimum and maximum comparison scores to be used for the sample pairs. The Minimum Score in the range must be between 0.0 and 99.9, and must be less than or equal to the Maximum Score. Note that Pairs are only returned for scores greater than or equal to the CR threshold configured during the bulk cross match. Default: 7</td>
</tr>
<tr>
<td>Maximum Score</td>
<td></td>
<td>The Score Range specifies the minimum and maximum comparison scores to be used for the sample pairs. The Maximum Score in the range must be between 0.0 and 99.9, and must be greater than or equal to the Minimum Score. Default: 25</td>
</tr>
</tbody>
</table>
| Result order    |              | Choices are Sorted by Score or Random.  
  • Sorted by score means that the distribution of scores across sample pair files is even. Each file will have a similar number of high-scoring matches and low-scoring matches.  
  • Random means the distribution of scores across sample pair files is completely random |
| Attributes to Return |            | Attributes you want to be included in the sample pairs file are selected on this tab.                                                                                                                     |
| Source Filter   |              | Sources you want to include are selected on this tab.                                                                                                                                                   |

Evaluating sample pairs:

The sample pairs file(s) contains pairs of members in color-coordinated rows that may be the same member or different members with similar attribute data. Someone familiar with the data, such as a data steward, must review the sample pairs and determine whether the members in each pair are the same or different.
About this task

When this job has completed, right-click the successful Generate Threshold Analysis Pairs job listed in the Jobs view and select Get job results. The Sample Pair Folder dialog opens, giving you the opportunity to specify the number of files across which to split the sample pairs data. Splitting the data across multiple files enables you to easily divide the evaluation task among multiple data stewards.

Procedure

1. Click Browse to select the path where you wish to save the sample pair file(s). This path can be any drive and directory accessible from the computer running IBM Initiate Workbench. The default location is the IBM Initiate Workbench workspace folder. To save the file(s) on the Hub server, use a mapped network drive to which the IBM Initiate Workbench computer user has write access.

2. In the Base File Name field, type the base file name to use for the sample pair file(s). If the File Count is greater than 1, the base file name will be appended with a digit. IBM Initiate Workbench appends all sample pair files with the .xls extension.

3. In the File Count field, select the number of files across which you wish to save the sample pair data. The maximum number of files is 25. The default is 1.

4. Click OK. A confirmation dialog shows the location and filename(s) used to save the sample pair data.

Results

This task can be done using the IBM Initiate Pair Manager application or a third-party spreadsheet application. The IBM Initiate Pair Manager provides an easy-to-use interface for selecting match status for the sample pairs. Refer to Chapter 12, “IBM Initiate Pair Manager,” on page 227 for instructions on using the application.

Note: The sample pair files may contain sensitive information, such as members’ social security numbers. Use the appropriate caution when distributing these files to data stewards for review.

Using a spreadsheet application to evaluate the sample pairs:

About this task

The next step in the weight generation process is to evaluate and adjust the Clerical Review Threshold and Autolink Threshold for each source combination and entity type. Instructions are found in the section titled Setting CR and AL thresholds” on page 173.

Procedure

1. If you choose to use a spreadsheet application rather than using the IBM Initiate Pair Manager, open the desired samplePairs.xls file. Sample pairs are banded with the same background color.

2. For each pair in the list, use the Same? column’s pull-down list to indicate whether the two records represent the same member (Yes) or different members (No). If you are unsure, select Maybe. For example, the members in rows 6 and 7 represent different people as evidenced by vastly different birth dates, although they have the same address. In the Same? column on either row 6 or row 7, you would select No. It is not necessary to set the Same? column on both rows for the pair.
3. When you are finished evaluating all the pairs in the sample file, save and close the file.

**What to do next**

The next step in the weight generation process is to evaluate and adjust the Clerical Review Threshold and Autolink Threshold for each source combination and entity type. Instructions are found in the section titled “Setting CR and AL thresholds” on page 173.

**Generate weights**

The Weight Generation tool calculates the weights of our data automatically using derived data as input.

This process can take several hours, depending on the size of the database, the complexity of the algorithm, and the number of attributes in the dictionary. For a dataset with 1,000,000 member records and 6-8 attributes, plan on a minimum of 2 hours processing time. Larger record sets will take more time to process, but the time estimate depends on multiple variables (e.g., hardware, dataset size, algorithm complexity, et cetera). For this reason, weights are usually generated at the end of the day so the utility can process data overnight and create less of a disruption for other applications that may be running on the same server. More information on the weight generation process is located in Chapter 6, “Weight generation,” on page 175. Generate Weights is run on the command line using the `MADCONFIG generate_weights` target.

**Note:** A new utility, mpxsmooth, is automatically run during the Generate Weights job to smooth weights. For information on the mpxsmooth utility, consult the *IBM Initiate Master Data Service Engine Installation Guide*.

**Table 4. Generate weights options**

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type</td>
<td>Specifies the entity type for which you wish to generate weights. All member types within the specified enttype are processed. If you are implementing multiple entity types (e.g., id for Identity and hh for Household), you will have to run generate weights separately for each type.</td>
</tr>
<tr>
<td>Weight generation steps</td>
<td>The following options comprise the weight generation process. Selecting an option executes that step; selecting the <strong>Execute all remaining steps through end of process</strong> option will also execute all remaining steps in the proper order. Typically, you will start your weight generation with the <strong>Delete artifacts from previous run</strong> step and enable <strong>Execute all remaining steps through end of process</strong>. Under certain circumstances, you might prefer to start with a step further down in the process.</td>
</tr>
<tr>
<td>Delete artifacts from previous run</td>
<td>Deletes all weight generation files that were created during a previous run of the Generate Weights job. Default: enabled</td>
</tr>
<tr>
<td>Workbench</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Generate counts for all attribute values</td>
<td>Counts the number of each attribute value represented in the data. This step executes the Generate Frequency Stats (mpxfreq) utility in -wgtmode. Default: disabled</td>
</tr>
<tr>
<td>Generate random pairs of members</td>
<td>Generates random pairs of members to be used in the comparison and matching process (for entity assignment). More information can be found in “Generating weights” on page 176. Default: disabled</td>
</tr>
<tr>
<td>Derive random data by comparing random members</td>
<td>Compares the random members for entity matching. Default: disabled</td>
</tr>
<tr>
<td>Perform matched candidate pairs reduction</td>
<td>The Hub attempts to narrow down the number of matched pairs to group as many members into a single entity as possible, as specified by the algorithm. Default: disabled</td>
</tr>
<tr>
<td>Generate matched set, matched statistics, and initial weights</td>
<td>Generate the matching and statistical data and initial weights that can be used for analytics and threshold calculations. Default: disabled</td>
</tr>
<tr>
<td>Skip last step because of too few attributes</td>
<td>If the data dictionary does not have enough attributes upon which to compare members, the Hub cannot make valid matching decisions. For example, if there are only two attributes such as name and address, to derive the matched statistics for name, we have only address. You cannot derive a good matched set from address alone. This option enables you to skip the Generate matched set, matched statistics and initial weights step while still performing the iteration step (when Execute all remaining steps through end of process is enabled).</td>
</tr>
<tr>
<td>Iterate over previous step and check for convergence of weights</td>
<td>Iterates through the data generated in the Generate matched set, matched statistics and initial weights step to check for convergence of weights.</td>
</tr>
<tr>
<td>Execute all remaining steps through end of process</td>
<td>Executes all steps below the selected option. Default: enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inputs and Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>BXM input directory</td>
</tr>
<tr>
<td>Workbench</td>
</tr>
<tr>
<td>Workbench</td>
</tr>
<tr>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>Number of random pairs bucket partitions</td>
</tr>
<tr>
<td>Default: 5</td>
</tr>
<tr>
<td>Number of matched pairs bucket partitions</td>
</tr>
<tr>
<td>Default: 5</td>
</tr>
<tr>
<td>Number of frequency partitions</td>
</tr>
<tr>
<td>Default: 10</td>
</tr>
<tr>
<td>Maximum number of input/output partitions</td>
</tr>
<tr>
<td>Number of random pairs to generate</td>
</tr>
<tr>
<td>Default: 3000000</td>
</tr>
<tr>
<td>Interval for reporting processed records</td>
</tr>
<tr>
<td>Maximum bucket set size</td>
</tr>
<tr>
<td>Minimum weight for writing item records</td>
</tr>
<tr>
<td>Options</td>
</tr>
<tr>
<td>Encoding</td>
</tr>
<tr>
<td>Default: latin1</td>
</tr>
<tr>
<td>Audrecno used for auditing</td>
</tr>
<tr>
<td>Default: 99</td>
</tr>
<tr>
<td>Comparison mode</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Log Options</td>
</tr>
<tr>
<td>Trace logging</td>
</tr>
<tr>
<td>Default: disabled</td>
</tr>
</tbody>
</table>
Table 4. Generate weights options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Description</th>
<th>Default: disabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug logging</td>
<td>Produces low-level diagnostics used internally by Initiate Systems to identify what was happening on the system before an error condition occurred. This option generates a large amount of output per activity and should only be used for short periods of time. <strong>Note:</strong> Debug logging can potentially include personal member information such as member identification number, name, etc.</td>
<td></td>
</tr>
<tr>
<td>Timer logging</td>
<td>Produces timings on certain operations to help identify where significant processing time is elapsing.</td>
<td>disabled</td>
</tr>
<tr>
<td>SQL logging</td>
<td>Outputs the SQL that is sent by the Master Data Engine database layer to the RDBMS. This helps in diagnosing database-related issues. This option can produce large amounts of output depending on the activity.</td>
<td>disabled</td>
</tr>
<tr>
<td>Audit logging</td>
<td>Produces activity information and non-critical warnings. Often, this option is used when a new system is first implemented to monitor activity.</td>
<td>disabled</td>
</tr>
<tr>
<td>Algorithm logging</td>
<td>A separate logging level for algorithm-related debug information without the risk of including protected health information (PHI).</td>
<td>disabled</td>
</tr>
</tbody>
</table>

When this job is complete, right-click the successful Generate Weights job listed in the Jobs view and select **Get job results**. Indicate the desired directory to which to save the weights. The default directory is the one originally specified on the initial job form, but you can specify a different directory prior to saving the weights if desired. The output of Generate Weights is then copied into the project from the Hub.

**Prepare Bucket Analysis**

Manages data for bucket analysis functions. If the number of records in the Hub is larger than 2 million, the bucket analysis queries will not execute unless the data is first “prepared” using the Bucket Analysis Preparation utility.

Data preparation involves taking the raw member and bucket data and pre-computing an intermediary set of data that can be quickly queried. Preparing data for 2-5 million records should take around 10 minutes, while preparing data for 50 million records will take approximately 5 hours. These estimates may vary wildly depending on different hardware and database configurations. If the member data is modified, then the prepared data should be recomputed to avoid seeing out-of-date results.
Table 5. Bucket analysis preparation options

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create bucket analysis data</td>
<td>Creates the bucket analysis data used to prepare analytics reports for large datasets.</td>
</tr>
<tr>
<td>Remove previously created bucket analysis data</td>
<td>Removes previously-created bucket analysis data. Performing this step ensures that the next time bucket analysis data is prepared, it does not contain out-of-date results.</td>
</tr>
</tbody>
</table>

Validate weights

Performs weight validation tasks. This job can be performed against weight table UNL files (created by a successful Generate Weights job) or directly against the Hub database. This utility is run on the command line using the MADCONFIG validate_weights target.

Table 6. Validate weights options

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type</td>
<td>This option identifies the type of entity being validated. If you are implementing multiple entity types (e.g., id for Identity and hh for Household), you will have to run Validate weights separately for each type. Default: (varies, depending on the available types)</td>
</tr>
</tbody>
</table>

Inputs and Outputs

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run heuristics against weight table UNLs</td>
<td>Specifies that the validation reports are to be run against the weight table UNL files generated on the Hub by the Generate Weights job. Default: enabled</td>
</tr>
<tr>
<td>Weight table UNL directory</td>
<td>Specifies the input directory in which the UNL files were created during Weight Generation. This directory is on the Hub relative to the project's work directory. This option is enabled only when ‘Run heuristics against weight table UNLs’ is also enabled. Default: unl</td>
</tr>
<tr>
<td>Run heuristics against weight tables in the hub database</td>
<td>Specifies that the validation tasks are to be run against the weight tables in the hub database, rather than against UNL files. Default: disabled</td>
</tr>
<tr>
<td>Report file format</td>
<td>The output format for the validation report. Valid options are xml and txt format. Default: xml</td>
</tr>
<tr>
<td>Encoding</td>
<td>Choices are Latin1, UTF8 and UTF16. Default: latin1</td>
</tr>
</tbody>
</table>

When this job is complete, right-click the successful Validate Weights job listed in the Jobs view and select Get job results. Indicate the desired directory to which to save the validate weights report. The default directory is the one originally specified on the initial job form, but you can specify a different directory prior to
saving the report if desired. The generated report is then copied into the
`entity_name` subdirectory of the specified project folder opens automatically in IBM
Initiate Workbench for review. The filename follows the convention
wgtest_results_yyyy-mm-ddThh-mm-ss.xml or wgtest_results_yyyy-mm-ddThh-mm-
ss.txt

Bulk Tools jobs

Most of the options for the data analysis jobs are self-explanatory. More
information is provided for select options where needed.

**Capture Same/Diff Rule for an Entity (mpxrule)**

Writes out Same and/or Diff rules for a given Entity type to .bin files. This job
needs to be run before mpxlink if you select the Same (identity) and/or Diff
(non-identity) option(s) in mpxlink.

The mpxrule job must be run prior to running mpxlink if you select the Same
(identity) and/or Diff (non-identity) option in mpxlink.

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type</td>
<td>-entType</td>
<td>This option identifies the type of entity being computed. If you are implementing multiple entity types (e.g., id for Identity and hh for Household), you will have to run Capture Same/Diff Rule for an Entity (mpxrule) for each type. Default: varies depending on available types</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Outputs</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output directory</td>
<td>-bxmOutDir</td>
<td>This is the directory in which you want the Capture Same/Diff Rule for an Entity (mpxrule) output binary files located. This is typically relative to the work directory on the server hosting the Hub configuration. Default: bxm</td>
</tr>
</tbody>
</table>

**Options**

<table>
<thead>
<tr>
<th>Options</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use explicit different records from entrule</td>
<td>-{no}bxmDiff</td>
<td>This option controls whether Link Entities (mpxlink) uses existing entity rules when forming entities. For example, if the users broke two members in an entity apart in Inspector, a non-identity rule is created by the engine (likewise if two members are manually linked, an “identity” rule is created). mpxrule captures these rules as “same” or “diff” rules (identity or non-identity, respectively). If the user is “re-crossmatching” an existing database, including these rules will prevent Link Entities (mpxlink) from reforming linkages (in the case of diff rules), or force members to be in the same entity (in the case of a “same” rule). The input data used here is created with the corresponding Use explicit different records from entrule (-bxmDiff) option in mpxcomp. Default: enabled</td>
</tr>
<tr>
<td>Workbench</td>
<td>Command line</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Use explicit same records from entrule</td>
<td>-(no)bxsSame</td>
<td>Like -bxmDiff, this option controls whether Link Entities (mpxlink) uses existing entity rules when forming entities. Refer to the description for -bxmDiff, above, for more details. The input data used here is created with the corresponding Use explicit same records from entrule (-bxsSame) option in mpxcomp. Default: enabled</td>
</tr>
</tbody>
</table>

**Log Options**

<table>
<thead>
<tr>
<th>Log Options</th>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace logging</td>
<td></td>
<td>Produces a trace of activity as interactions flow through the system. This option is very verbose and should only be used for short periods of time. Default: disabled</td>
</tr>
<tr>
<td>Debug logging</td>
<td></td>
<td>Produces low-level diagnostics used internally by Initiate Systems to identify what was happening on the system before an error condition occurred. This option generates a large amount of output per activity and should only be used for short periods of time. <strong>Note:</strong> Debug logging can potentially include personal member information such as member identification number, name, etc. Default: disabled</td>
</tr>
<tr>
<td>Timer logging</td>
<td></td>
<td>Produces timings on certain operations to help identify where significant processing time is elapsing. Default: disabled</td>
</tr>
<tr>
<td>SQL logging</td>
<td></td>
<td>Outputs the SQL that is sent by the Master Data Engine database layer to the RDBMS. This helps in diagnosing database-related issues. This option can produce large amounts of output depending on the activity. Default: disabled</td>
</tr>
<tr>
<td>Audit logging</td>
<td></td>
<td>Produces activity information and non-critical warnings. Often, this option is used when a new system is first implemented to monitor activity. Default: disabled</td>
</tr>
<tr>
<td>Algorithm logging</td>
<td></td>
<td>A separate logging level for algorithm-related debug information without the risk of including protected health information (PHI). Default: disabled</td>
</tr>
</tbody>
</table>

**Compare Members in Bulk (mpxcomp)**
The Compare Members in Bulk (mpxcomp) utility enables the comparison of records and is one of the processes used during bulk (BXM) and incremental cross matches (IXM).
When run, this utility selects candidates, compares member records, and assigns comparison scores. mpxcomp must be run once for each type of entity (e.g., identity and household) implemented, as the comparison algorithm is specific to each entity type.

With regard to system performance, note that mpxcomp loads the entire input data set into memory for processing. If you are working with large files, this may cause memory issues, because your machine must have sufficient continuous memory to accommodate the data files. For large data sets, you may elect to use the *Part options to conserve system memory and optimize performance. Use of these options (-nMemParts, -nBktParts, -minBktPart, -maxBktPart and -maxParts) partitions the data to avoid pulling the entire set into memory at one time. BktParts should be the first option adjusted to accommodate available memory.

If you plan to partition data, a partitioning strategy should be devised before beginning data derivation. Data must be partitioned consistently between the derivation step (mpxdata, mpxfsdvd, mpxprep, mpxredvd), the comparison step (mpxcomp), and the linking step (mpxlink). In addition to the derived data binary files, you must have the following in place before running mpxcomp:

- a Master Data Engine instance if you are running the utility from IBM Initiate Workbench; if running from a command line, the Engine instance is not required
- Hub configured with your algorithm and data dictionary (includes threshold settings)
- Weights

The BXM process uses the weights to create an aggregate comparison score, which is then compared to the threshold settings to determine auto-links and tasks.

The output is additional binary files that represent the entity link and task groupings and comparison scores. This output is the input to the next phase of BXM, which is Link Entities (mpxlink).

**When to run Compare Members in Bulk (mpxcomp)**

You will run Compare Members in Bulk (mpxcomp):

- during the initial stage of implementation to generate baseline comparison scores.
- during the "reiterate" step of the implementation process (refer to your Bootcamp materials for information on the reiterate step). After going through the entire set of implementation steps and analyzing your data results, you may determine that modifications to your algorithm and data dictionary are necessary. If so, you will typically re-derive your data (via mpxdata, mpxfsdvd, or mpxredvd) and run another BXM.
- after implementation if you modify the attributes that are used by your comparison functions (e.g., adding a alias to a name comparison) or changes to your bucketing configuration. Comparison function and bucket changes require new weights, re-derivation of data and a new BXM.
- when running an IXM.
## Table 7. Compare Members in Bulk (mpxcomp) options

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type</td>
<td>-entType</td>
<td>This option identifies the type of entity being computed. If you are implementing multiple entity types (e.g., id for Identity and hh for Household), you will have to run Compare Members in Bulk (mpxcomp) for each type. Default: varies depending on available types</td>
</tr>
<tr>
<td>Inputs and Outputs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input directory</td>
<td>-bxmInpDir</td>
<td>This is the directory where the input binary (.bin) files from Derive Data and Create UNLs (mpxdata) are stored. This is typically the work directory created under hub_instance_path/inst/ mpinet_hub_instance_name/work. Note that multiple Hubs might share the same hub_instance_path dir, but will have different hub_instance_name. The Hub instance that you connect to will dictate where the work directory lives. You can list multiple directories for this option; separate multiple directories with single spaces. Default: bxm</td>
</tr>
<tr>
<td>Output directory</td>
<td>-bxmOutDir</td>
<td>This is the directory in which you want the Compare Members in Bulk (mpxcomp) output binary files located. This is typically relative to the work directory on the server hosting the Hub configuration. Default: bxm</td>
</tr>
<tr>
<td>Workbench</td>
<td>Command line</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Performance tuning |              | The term “partitions” as used in these options refers to breaking the member, bucket or query data files into pieces. The derivation utilities (mpxdata, mpxprep, mpxfsdvd, mpxredvd) produce a set of initial “BXM” files that are used by other utilities down-stream to do a cross match or generate weights. If the data set is large, the BXM files will subsequently be large. The utilities that read these files (e.g., mpxcomp) need to be able to fit this data into the available memory (RAM). If the memory requirement is larger than the available memory, the processes may swap or even run out of memory and fail. By breaking the data into pieces (partitions), the utilities can read pieces of the BXM data at a time and run within the available memory.  
  
  **Note:**  
  1. Both BktParts and MemParts options are specified when doing data derivation. The output of the DVD utilities become the input for Compare Members in Bulk (mpxcomp). The value specified for mpxcomp should match the value set for the data derivation processes.  
  2. minBktPart and maxBktPart settings override any value set for nBktParts.  
  3. To expedite the BXM process, set minBktPart and maxBktPart to run multiple processes against the same pool of buckets part files. For example, with a pool of ten files (mpx_bxmbktd.001 through mpx_bxmbktd.010), running the Compare Members in Bulk (mpxcomp) set with -minBktPart 5 and -maxBktPart 10 instructs mpxcomp to consume only mpx_bxmbktd.005 through mpx_bxmbktd.010.                                                                                     |
| Number of threads | -nthreads    | The number of threads to used for the Compare Members in Bulk (mpxcomp) process. The number of threads set can have an affect on system performance. The value should correspond to the number of CPU’s available on the machine (e.g., if the machine has 4 CPU’s, set the number of threads for mpxcomp to 4 to optimize the time it takes to process).  
  
  Default: 1  
  Maximum value: 64  
  Recommended value: 1 thread per processor |
**Table 7. Compare Members in Bulk (mpxcomp) options (continued)**

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of member partitions</td>
<td>-nMemParts</td>
<td>This option identifies the data partitions consumed by Compare Members in Bulk (mpxcomp). When this option is defined for mpxdata, mpxprep, mpxredvd or mpxfsdvd, the processes will break up the data from memHead and memCmpd (comparison data). While using this option can cut your memory usage significantly, your setting can impact performance. The higher the memParts setting, the slower your comparison process simply because the Engine is forced to do more duplicate comparisons. In order to compare every member that shares at least one bucket, the Engine compares each memPart against itself and then against all other memParts. For example if you had memParts set to 3, you would have parts A, B, and C. For each BktPart you would compare: A &gt; A, A &gt; B, A &gt; C B &gt; B, B &gt; C C &gt; C If it is necessary to use this option, specify a minimum setting of 3. Since the comparison has to bring in two parts to compare against each other, only splitting the data in half does not save memory. You should always use just enough parts to get all of the comparison data into physical memory. Default: 1 Maximum value: 100 Recommended value: 1. The value used for the data derivation process should be the same values used for Compare Members in Bulk (mpxcomp) and Link Entities (mpxlink).</td>
</tr>
<tr>
<td>Workbench</td>
<td>Command line</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>--------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Number of bucket partitions</td>
<td>-nBktParts</td>
<td><strong>This option breaks up the member bucket data (membktld) into smaller, more manageable chunks to optimize memory use.</strong> This option can assist sort performance on large data sets. All members that share a given bucket value will end up in the same part and be compared to one another. The number of bucket parts you set when running Compare Members in Bulk (mpxcomp) should match the number you specified for the derivation process (mpxdada, mpxfsdvd, or mpxredvd). If running an IXM, set this to the number used in the original data load. BktParts should be the first option adjusted to accommodate available memory. <strong>Note:</strong> If -minBktPart and -maxBktPart options in Compare Members in Bulk (mpxcomp) are used, they will override any settings for -nBktParts. When attempting to reduce your memory footprint, increase nBktParts before adjusting MemParts. Running a utility with the -noexec option outputs memory usage requirements that can help you determine how to adjust the -n*Part settings. Default: 10 Maximum value: 100 Recommended value: Should match the number of bktParts created in the bxmInpDir.</td>
</tr>
<tr>
<td>Minimum bucket partitions to process</td>
<td>-minBktPart</td>
<td><strong>Determines the minimum number of bucket parts to process.</strong> Both -minBktPart and -maxBktPart are performance options that allow Compare Members in Bulk (mpxcomp) to process a range of bucket parts. This is often used when running Compare Members in Bulk (mpxcomp) across multiple machines. For example, you can run bucket parts 1 through 5 on machine 1 and parts 6 through 10 on machine 2. Default: 0 Maximum value: Less than or equal to the value of maxBktPart</td>
</tr>
<tr>
<td>Maximum bucket partitions to process</td>
<td>-maxBktPart</td>
<td><strong>Determines the maximum number of bucket parts to process.</strong> Default: 0 Maximum value: 100</td>
</tr>
<tr>
<td>Workbench</td>
<td>Command line</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Maximum number of output partitions | -nMxmParts    | This option partitions the output of Compare Members in Bulk (mpxcomp) into smaller chunks for use by Link Entities (mpxlink). The value set for mpxcomp determines how many partitioned file segments are passed to Link Entities (mpxlink), thus the MxmParts value for both must be the same.  
  Default: 1  
  Maximum value: 100  
  **Note:** mpxcomp will fail when:  
  - Either minBktPart or maxBktPart is set to 0 (which means bktPart is not used). The min/max bucket parts must be set to a valid range (e.g., -minBktPart 1 - maxBktPart 5).  
  - The value of minBktPart is greater than or equal to the value of maxBktPart |
| Maximum bucket set size       | -maxbktsize   | Maximum bucket size determines the maximum number of members that may have the same bucket value for candidate selection. For example, using the default of 500, of more than 500 members have the same bucket value, Compare Members in Bulk (mpxcomp) will ignore those members for comparison. The value set depends on a variety of factors, including the number of members in the database and the bucketing strategy. The log will report bucket hash values exceeding this parameter, as well as 5 members for the user to examine. If set appropriately, the values reported in the log may indicate a bucket value that should be defined as an anonymous value.  
  Default: 500  
  Maximum value: 1048576 |
| Options                       |                |                                                                                                                                                                                                             |
| Encoding                      | -encoding      | Determines the encoding of the .unl files. Options are Latin1, UTF8, and UTF16.  
  Default: Latin1               |
| Minimum bucket role           | -minBktTag     | Minimum bucket tag to use. Bucket tags are used for speed optimization and allow bucket data to be created on bucket roles greater than or equal to the minimum bucket tag and less than or equal to the maximum bucket tag. Using the bucket tag option enables you to eliminate roles that do not have any impact on the linking outcome.  
  Default: 0  
  Maximum value: 15               |
| Maximum bucket role           | -maxBktTag     | Maximum bucket tag to use.  
  Default: 0  
  Maximum value: 15               |
Table 7. Compare Members in Bulk (mpxcomp) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write linkage item records</td>
<td>-{no}bxmLink</td>
<td>Determines whether to write output for linkage records. Output is written to a file. Disable this option if you do not want to write records.</td>
</tr>
<tr>
<td>Default: -bxmLink, (write linkage records)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write task item records</td>
<td>-{no}bxmTask</td>
<td>Determines whether to write output for task records. Disable this option if you do not want to write records.</td>
</tr>
<tr>
<td>Default: -bxmTask, (write task records)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write review identifier item records</td>
<td>-{no}bxmRvid</td>
<td>Determines whether to write output for Review Identifier task records. Disable this option if you do not want to write records.</td>
</tr>
<tr>
<td>Default: -bxmRvid, (write Review Identifier tasks)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do only different source comparisons</td>
<td>-{no}DiffSrcOnly</td>
<td>Use this option if you want to compare members from one source only to records in a different source. For example, records from Source A will be compared to those in Source B and C, but not against other records in A.</td>
</tr>
<tr>
<td>Default: -noDiffSrcOnly, (compares records across all sources)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do only same source comparisons</td>
<td>-{no}SameSrcOnly</td>
<td>Use this option to compare records only against those from the same source. For example, records from Source A are compared against A, but not against B and C.</td>
</tr>
<tr>
<td>Default: -noSameSrcOnly, (compares records across all sources)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create a dense memhead lookup table</td>
<td>-{no}dense</td>
<td>When this option is enabled (-dense), Compare Members in Bulk (mpxcomp) creates a memhead lookup table that is used during the comparison operation. The lookup table replaces runtime computation with a simple array indexing operation. The -dense option uses more memory, but is faster than -nodense. Only specify -dense when you have sufficient memory for the dataset and if you have large gaps in your memRecno ranges.</td>
</tr>
<tr>
<td>Default: -nodense</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enable incremental cross match</td>
<td>-ixmMode</td>
<td>Use this option to enable incremental cross matching. In IXM mode, a subset of members are compared rather than the entire member set. If running a BXM, use the default of false. If running an IXM, set this to true.</td>
</tr>
<tr>
<td>Default: disabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generate memory usage information only</td>
<td>-</td>
<td>The memory usage information generated by this option is viewable in IBM Initiate Workbench by executing the &quot;Get job results&quot; action on the Compare Members in Bulk (mpxcomp) log job returned by the Hub.</td>
</tr>
<tr>
<td>Default: disabled</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7. Compare Members in Bulk (mpxcomp) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparison mode</td>
<td>-cmpMode</td>
<td>This option controls the Compare Members in Bulk (mpxcomp) comparison behavior and is intended to improve performance by excluding comparisons configured only for searching. Comparison modes can be set in your algorithm as follows: cmpmode 1 = match and link members, cmpmode 2 = search members, cmpmode 3 = search, match and link members. The mode set in your algorithm does not have to match the option specified for mpxcomp. The Compare Members in Bulk (mpxcomp) option acts as a filter for selecting which comparison functions will be used for comparison. For example, if you specify match and link (option 1), mpxcomp uses only the comparison functions that are set to match and link. Typically you would use Compare Members in Bulk (mpxcomp) with Match, link and search (comparison mode 3). If this option is not set, all comparison modes configured in your algorithm are compared. If set to Match and link only, comparison modes 1 and 3 are compared. If set to Search only, comparison modes 2 and 3 are compared. Default: Match, link and search (comparison mode 3)</td>
</tr>
</tbody>
</table>

Log Options

<table>
<thead>
<tr>
<th>Trace logging</th>
<th>Produces a trace of activity as interactions flow through the system. This option is very verbose and should only be used for short periods of time. Default: disabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug logging</td>
<td>Produces low-level diagnostics used internally by Initiate Systems to identify what was happening on the system before an error condition occurred. This option generates a large amount of output per activity and should only be used for short periods of time. Note: Debug logging can potentially include personal member information such as member identification number, name, etc. Default: disabled</td>
</tr>
<tr>
<td>Timer logging</td>
<td>Produces timings on certain operations to help identify where significant processing time is elapsing. Default: disabled</td>
</tr>
<tr>
<td>SQL logging</td>
<td>Outputs the SQL that is sent by the Master Data Engine database layer to the RDBMS. This helps in diagnosing database-related issues. This option can produce large amounts of output depending on the activity. Default: disabled</td>
</tr>
</tbody>
</table>
Table 7. Compare Members in Bulk (mpxcomp) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit logging</td>
<td>Produces activity information and non-critical warnings. Often, this option is used when a new system is first implemented to monitor activity. Default: disabled.</td>
<td></td>
</tr>
<tr>
<td>Algorithm logging</td>
<td>A separate logging level for algorithm-related debug information without the risk of including protected health information (PHI). Default: disabled.</td>
<td></td>
</tr>
</tbody>
</table>

Derive Data and Create UNLs (mpxdata)

The Derive Data and Create UNLs (mpxdata) utility performs several steps while running, from parsing data into UNL files to deriving data and organizing member records into buckets. The Derive Data and Create UNLs (mpxdata) utility also creates binary files, which are used to compare data faster than scanning through strings. Derive Data and Create UNLs (mpxdata) parses raw data extracts into Attribute-specific sets of data, for example taking a single record for a person and creating one record for the SSN, another for the name elements, and a third for the telephone numbers. This allows the Hub to store multiple iterations of active and inactive data (such as a former address or phone number) and increases responsiveness when searching and comparing.

A configuration file is required for this job. Details on building a configuration file are located in Appendix E, “MPXDATA configuration file,” on page 353.

Table 8. Derive Data and Create UNLs (mpxdata) options

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Type</td>
<td>-memType</td>
<td>This applies only to MEMCOMPUTE mode. This option sets a filter on the output of Derive Data and Create UNLs (mpxdata) for the specified member type. Setting this field to ALL will process all member types for the Hub.</td>
</tr>
</tbody>
</table>

Input

<table>
<thead>
<tr>
<th>Input</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input file</td>
<td>-inpFile</td>
<td>The extract filename that contains the data, either fixed length or delimited by the Input File Format’s Field delimiter. The location of this file is on the Hub, relative to the instance directory’s project workspace: (hub_instance_path\inst\mpinet_hub_instance_name\work\projectname\work).</td>
</tr>
<tr>
<td>Number of header rows in input file to skip</td>
<td>-skipRecs</td>
<td>If there are any rows of text in the input file before the actual data rows begin, indicate how many rows to ignore. This does not include lines commented out with the # character.</td>
</tr>
<tr>
<td>Input File Format</td>
<td>-fldDelim</td>
<td>Delimited: This option is for extract files with variable-length record files. • Field delimiter: Field delimiter character for variable length record fields.</td>
</tr>
</tbody>
</table>
Table 8. Derive Data and Create UNLs (mpxdata) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input File Format</td>
<td>-recSize</td>
<td>Fixed: For extract files with fixed-length record fields, the length field must be specified. See Appendix F, “MPXDATA configuration file,” on page 353.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Record length: The size of each record (for fixed-length input files). Add the appropriate end-of-line characters to this value.</td>
</tr>
<tr>
<td>Input Config</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Import From File</td>
<td>-config</td>
<td>Use this button to specify the name of a configuration file. This is a specially formatted file defining the fields for the data input file. For more information on the structure of the configuration file, consult Appendix F, “MPXDATA configuration file,” on page 353.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add</td>
<td></td>
<td>Use this button to manually add fields describing the data input file.</td>
</tr>
<tr>
<td>Outputs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generate UNL output</td>
<td>-unlOutDir</td>
<td>This applies only to MEMCOMPUTE mode. This option instructs Derive Data and Create UNLs (mpxdata) to create UNL files after reading and parsing the extract file. Default: enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and -unlOutSegs used together</td>
</tr>
<tr>
<td>UNL output directory</td>
<td>-unlOutDir</td>
<td>This applies only to MEMCOMPUTE mode. Specifies the directory where the UNL files should be saved. Default: unl</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generate bucket UNL</td>
<td>use -unSegList with</td>
<td>Instructs Derive Data and Create UNLs (mpxdata) to create UNL files containing bucketing data. Default: enabled</td>
</tr>
<tr>
<td></td>
<td>-unlOutDir</td>
<td></td>
</tr>
<tr>
<td>Generate comparison UNL</td>
<td>use -unSegList with</td>
<td>Instructs Derive Data and Create UNLs (mpxdata) to create UNL files containing comparison data. Default: enabled</td>
</tr>
<tr>
<td></td>
<td>-unlOutDir</td>
<td></td>
</tr>
<tr>
<td>Generate query UNL</td>
<td>use -unSegList with</td>
<td>This option is for use with the relationship linker and instructs Derive Data and Create UNLs (mpxdata) to create UNL files containing query data. The relationship types, attributes and rules should already be defined so that Derive Data and Create UNLs (mpxdata) knows what data to include in the UNL file. Default: disabled</td>
</tr>
<tr>
<td></td>
<td>-unlOutDir</td>
<td></td>
</tr>
<tr>
<td>Generate BXM output</td>
<td>-(no)bxmCmpd and</td>
<td>This option instructs Derive Data and Create UNLs (mpxdata) to create BXM files after reading and parsing the extract file. Default: enabled</td>
</tr>
<tr>
<td></td>
<td>-(no)bxmBktd</td>
<td></td>
</tr>
<tr>
<td>BXM output directory</td>
<td>-bxmOutDir</td>
<td>This applies only to MEMCOMPUTE mode. Specifies the directory where the BXM files should be saved. Default: bxm</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate query BXM</td>
<td><code>-{no}bxmlQryd</code></td>
<td>This option is for use with the relationship linker and instructs Derive Data and Create UNLs (mpxdata) to create BXM files containing query data. The relationship types, attributes and rules should already be defined so that Derive Data and Create UNLs (mpxdata) knows what data to include in the BXM file. Default: disabled</td>
</tr>
<tr>
<td>Rejects file</td>
<td><code>-rejFile</code></td>
<td>If Derive Data and Create UNLs (mpxdata) is unable to parse data as it reads each row in the input file, it writes that data to the Rejects file. The Derive Data and Create UNLs (mpxdata) utility will continue to parse remaining data, adding any additional 'rejected' data to the rejects file. Default: rejects.txt</td>
</tr>
<tr>
<td>Append to existing data</td>
<td><code>-append</code></td>
<td>This option only applies to the UNL files. If you are processing multiple extract data files (e.g., from different sources), Derive Data and Create UNLs (mpxdata) will write new (or overwrite existing) unl files when this option is unchecked. If checked, the new UNL data written by Derive Data and Create UNLs (mpxdata) is added to the end of the existing UNL file. Default: disabled</td>
</tr>
<tr>
<td>Performance Tuning</td>
<td></td>
<td>The term “partitions” as used in these options refers to breaking the member, bucket or query data files into pieces. The derivation utilities (mpxdata, mpxprep, mpxfsdvld, mpxredvd) produce a set of initial &quot;bxm&quot; files that are used by other utilities down-stream to do a cross match or generate weights. If the data set is large, the bxm files will subsequently be large. The utilities that read these files (e.g., mpxcomp) need to be able to fit this data into the available memory (RAM). If the memory requirement is larger than the available memory, the processes may swap or even run out of memory and fail. By breaking the data into pieces (partitions), the utilities can read pieces of the bxm data at a time and run within the available memory.</td>
</tr>
<tr>
<td>Maximum number of Member partitions</td>
<td><code>-nMemParts</code></td>
<td>Applies only to MEMCOMPUTE mode. Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the Hub. The utility that consumes the Derive Data and Create UNLs (mpxdata) output (such as Generate Frequency Stats (mpxfreq)) must use a matching “memparts” value. A good rule of thumb is to leave this at the default unless you need the memory. The higher the member partitions the slower your Compare Members in Bulk (mpxcomp) process, as the Hub must do more duplicate comparisons. Default: 1</td>
</tr>
<tr>
<td>Workbench</td>
<td>Command line</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Maximum number of Bucket partitions</td>
<td>-nBktParts</td>
<td>Applies only to MEMcompute mode. Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the Hub. A good rule of thumb is to leave this at the default unless you need the memory. Default: 10</td>
</tr>
<tr>
<td>Maximum number of Query partitions</td>
<td>-nQryParts</td>
<td>Applies only to MEMcompute mode. Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the Hub. A good rule of thumb is to leave this at the default unless you need the memory. This option is enabled only when the option to Generate query BXM is also enabled. Default: 1</td>
</tr>
<tr>
<td>Buffer size</td>
<td>-buffSize</td>
<td>Size for each file I/O buffer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 65536</td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mode</td>
<td>-ixnCode</td>
<td>Choices are MEMcompute or MEMput. MEMput does inserts or updates to members in an existing Hub database for each record in the input file. MEMcompute generates UNL files which can then be loaded using load UNLs to DB (madunlload), or another load utility. In most cases when processing an extract, MEMcompute is used, because loading UNL files is much faster than inserting each member via MEMput. Default: MEMcompute</td>
</tr>
<tr>
<td>Maximum errors before stopping</td>
<td>-maxErrs</td>
<td>This option sets a threshold for errors in the data. Once the threshold is reached, Derive Data and Create UNLs (mpxdata) stops (but that is not considered an mpxdata error). The intent of this option is to allow you to process an extract with tolerance for known data issues. For example, if the delimited extract file has too few or too many delimiters in a few records, you can set this option to an expected value and if it is exceeded, Derive Data and Create UNLs (mpxdata) will stop, giving you an opportunity to resolve the problem in the extract data or config file. The Derive Data and Create UNLs (mpxdata) utility writes records it cannot parse to the rejects file, and the total error count (totErrs) is reported in the mpxdata log as an INFO message: 06:59:53 mpxdata INFO MPX_BxmData: totRecs=6, totErrs=3, elapsed=1 secs., recs/sec=6, minbkttag=0, maxbkttag=0, nMemParts=1, nBktParts=1, buffsize=65536 Default: 100</td>
</tr>
<tr>
<td>Workbench</td>
<td>Command line</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Don’t stop on errors</td>
<td></td>
<td>If checked, Derive Data and Create UNLs (mpxdata) will not halt processing when it encounters a data error. It does, however, write records it cannot process to the rejects file. Checking this option disables the 'Maximum errors before stopping' option. Default: disabled</td>
</tr>
<tr>
<td>Maximum records to process</td>
<td>-maxRecs</td>
<td>Indicates the maximum number of records to process before ending the Derive Data and Create UNLs (mpxdata) process. This option is disabled when the Process all records option is checked. Default: 0 (process all records)</td>
</tr>
<tr>
<td>Process all records</td>
<td></td>
<td>Enabled by default to instruct Derive Data and Create UNLs (mpxdata) to process all records in the input file. Unchecking this option enables the 'Maximum records to process' option. Default: enabled</td>
</tr>
<tr>
<td>Processed record report interval</td>
<td>-rptRecs</td>
<td>The Derive Data and Create UNLs (mpxdata) log reports a status every $n$ records. You may wish to decrease the frequency to reduce the log output for very large datasets, or increase it to get more granularity. Default: 100,000</td>
</tr>
<tr>
<td>Starting memrecno</td>
<td>-memRecno</td>
<td>This applies only to MEMCOMPUTE mode. The value supplied is used as the first memrecno in the UNL files and incremented by one for each additional record. Default: 1</td>
</tr>
<tr>
<td>Starting audrecno</td>
<td>-audRecno</td>
<td>This applies only to MEMCOMPUTE mode. The value supplied is used as the first audrecno in the UNL files and incremented by one for each additional record. Default: 2</td>
</tr>
<tr>
<td>Minimum bucket role</td>
<td>-minBktTag</td>
<td>This applies only to MEMCOMPUTE mode and specifies the lowest bucketing role to be included in the operation. Default: 0</td>
</tr>
<tr>
<td>Maximum bucket role</td>
<td>-maxBktTag</td>
<td>This applies only to MEMCOMPUTE mode and specifies the highest bucketing role to be included in the operation. Default: 0 (include all)</td>
</tr>
<tr>
<td>Minimum query role</td>
<td>-minQryRole</td>
<td>This applies to MEMCOMPUTE mode when the 'Generate query BXM' option is enabled. This option specifies the lowest query role to be included in the operation. Default: 10,000</td>
</tr>
</tbody>
</table>
### Table 8. Derive Data and Create UNLs (mpxdata) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoding</td>
<td>-encoding</td>
<td>Choices are Latin1, UTF8 and UTF16. Default: Latin1</td>
</tr>
</tbody>
</table>
| Mem mode  | -memMode     | This applies only to MEMPUT mode. Choices are complete, partial, attrcomp and explicit. More detail on these modes can be found in the “Member put interaction” appendix of the *IBM Initiate Master Data Service SDK Reference for Java and Web Services*.  
  **Partial**  
  is used when a source system sends an update to a member, but you do not know if this is a complete picture of the member, or if you have the complete range of values for a given attribute.  
  **Attrcomp**  
  stands for attribute complete. Like the partial mode, the attrcomp mode tells the software that it may not have a complete picture of all the attributes that make a complete member, but for the attributes that are present, all known values for the member are included in the member put interaction.  
  **Complete**  
  tells the engine that the input to the member put interaction contains all of the values for all of the attributes defined for this member type.  
  **Explicit**  
  is used in situations where you want to control exactly what is stored and the record status of the attributes being stored. Default: Complete |
| Put type   | -putType     | This applies only to MEMPUT mode. Choices are insert_update, insert_only and update_only. This option works at a member level, not an attribute level.  
  **insert_only**  
  restricts the Master Data Engine to creating a new member. If a member already exists for this srcCode/memIdnum, the interaction will fail with an error code of EXISTS.  
  **update_only**  
  restricts the Master Data Engine to updating existing members only. If an attempt is made to update a member that does not already exist, the interaction will fail with an error code of ENOREC.  
  **insert_update**  
  adds a member if one does not exist, or it will update the existing member if it does exist. Default: insert_update |
<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Entity Priority</td>
<td>-entPrior</td>
<td>In IBM Initiate Workbench, you can specify a Default Entity Priority for each definitional source. Another option for entity management priority is to set the priority at the time of the member write (such as when executing Derive Data and Create UNLs (mpxdata)). This enables you to set a lower priority for batch loaded members regardless of the associated source. The default setting for Set Entity Priority is 0, which instructs the entity manager to use the source’s Default Entity Priority setting.</td>
</tr>
<tr>
<td>Write Audhead records</td>
<td>-audhead</td>
<td>This applies only to MEMCOMPUTE mode. When enabled, audhead records are written to UNL files (to be uploaded to the database later). Default: disabled</td>
</tr>
<tr>
<td>Attribute Types</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute type list</td>
<td>-un10utSegs</td>
<td>Enables you to select the attributes (segments) to be included in the UNL output files.</td>
</tr>
<tr>
<td>Log Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trace logging</td>
<td></td>
<td>Produces a trace of activity as interactions flow through the system. This option is very verbose and should only be used for short periods of time. Default: disabled</td>
</tr>
<tr>
<td>Debug logging</td>
<td></td>
<td>Produces low-level diagnostics used internally by Initiate Systems to identify what was happening on the system before an error condition occurred. This option generates a large amount of output per activity and should only be used for short periods of time. Note: Debug logging can potentially include personal member information such as member identification number, name, etc. Default: disabled</td>
</tr>
<tr>
<td>Timer logging</td>
<td></td>
<td>Produces timings on certain operations to help identify where significant processing time is elapsing. Default: disabled</td>
</tr>
<tr>
<td>SQL logging</td>
<td></td>
<td>Outputs the SQL that is sent by the Master Data Engine database layer to the RDBMS. This helps in diagnosing database-related issues. This option can produce large amounts of output depending on the activity. Default: disabled</td>
</tr>
<tr>
<td>Audit logging</td>
<td></td>
<td>Produces activity information and non-critical warnings. Often, this option is used when a new system is first implemented to monitor activity. Default: disabled</td>
</tr>
</tbody>
</table>
Table 8. Derive Data and Create UNLs (mpxdata) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm logging</td>
<td></td>
<td>A separate logging level for algorithm-related debug information without the risk of including protected health information (PHI). Default: disabled</td>
</tr>
</tbody>
</table>

When this job is complete, right-click the successful Derive Data and Create UNLs (mpxdata) job listed in the Jobs view and select Get job results. The Get job results action retrieves the job log file, shown on the Log Output tab, as well as the rejects file, shown on the Rejected Data tab. Note that retrieving these job files does not save any data or files to the IBM Initiate Workbench project.

**Derive Data from Hub (mpxredvd)**

Derive Data from Hub (mpxredvd) can be used both to derive member data (creating mpi_memcpd.unl and mpi_mekbkd.unl files), and (optionally) to create binary data files for use in a BXM. To produce these files, Derive Data from Hub (mpxredvd) reads the member data from the database (specifically, from the MEM tables), rather than from .unl files. Derive Data from Hub (mpxredvd) goes through the data line by line to create new buckets and comparison strings. You can select the specific elements (buckets, comparison strings, or binaries) you want to re-derive.

Table 9. Derive Data from Hub (mpxredvd) options

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Type</td>
<td>-memType</td>
<td>If you have multiple member types in the Hub database and only need to derive data for one of those member types, select the desired member type here; otherwise, select ALL. Default: ALL</td>
</tr>
<tr>
<td>Inputs and Outputs</td>
<td>-unlOutDir and -unlOutSegs used together</td>
<td>Indicates whether Derive Data from Hub (mpxredvd) should generate UNL files during processing. Default: enabled</td>
</tr>
</tbody>
</table>
### Table 9. Derive Data from Hub (mpxredvd) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
</table>
| UNL output directory   | -unlOutDir       | The output of Derive Data from Hub (mpxredvd) is the derived data segments (comparison, bucket and, optionally, query data), which have their own unl files (mpi_memcmpd, mpi_membktd, and mpi_memqryd, respectively) written to the directory specified here. This directory is relative to the project's work directory on the Hub:  

```
hub_instance_path\inst\mpinet_hub_instance_name\work\projectname\work\UNL_output_dir
```

With `\unlOutSegs`, indicates whether Derive Data from Hub (mpxredvd) should generate UNL files during processing. Also with `\unlOutSegs`, creates UNL files containing bucketing data or comparison data, or instructs mpxredvd to generate query UNL files during processing (the files are used by the relationship linker).

Default: unl

<table>
<thead>
<tr>
<th>Generate Bucket UNL</th>
<th>use -unlOutSegs</th>
<th>Instructs Derive Data from Hub (mpxredvd) to create UNL files containing bucketing data.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>with -unlOutdir</td>
<td>Default: enabled</td>
</tr>
<tr>
<td>Generate Comparison UNL</td>
<td>use -unlOutSegs</td>
<td>Instructs Derive Data from Hub (mpxredvd) to create UNL files containing comparison data.</td>
</tr>
<tr>
<td></td>
<td>with -unlOutdir</td>
<td>Default: enabled</td>
</tr>
<tr>
<td>Generate Query UNL</td>
<td>use -unlOutSegs</td>
<td>Instructs Derive Data from Hub (mpxredvd) to generate query UNL files during processing. These files are used by the relationship linker.</td>
</tr>
<tr>
<td></td>
<td>with -unlOutdir</td>
<td>Default: disabled</td>
</tr>
<tr>
<td>Generate BXM output</td>
<td>-{no}bxmCmpd</td>
<td>Instructs Derive Data from Hub (mpxredvd) to generate output files for bulk cross matching.</td>
</tr>
<tr>
<td></td>
<td>and -{no}bxmBktd</td>
<td>Default: enabled</td>
</tr>
</tbody>
</table>
| BXM output directory  | -bxmOutDir       | Indicates where you want the .bxm output files to be located. This directory is relative to the project's work directory on the Hub:  

```
hub_instance_path\inst\mpinet_hub_instance_name\work\projectname\work\BXM_output_dir
```

Default: bxm

| Generate query BXM    | -{no}bxmQryd    | Indicates whether Derive Data from UNLs (mpxfsdvd) should generate query BXM files during processing. These files are used by the relationship linker.                                                                                                                                                                                                                                                                            |
|                       |                 | Default: disabled                                                                                                                                                                                                                                                                                                                                                                                                           |
| Update database       | -{no}dbUpdate   | Updates the Hub database after the rederivation is completed.                                                                                                                                                                                                                                                                                                                                                               |
### Table 9. Derive Data from Hub (mpxredvd) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td></td>
<td>These fields act as a filter to include buckets up to the maximum (maximum bucket role), above a minimum (minimum bucket role) or within a range if both are set to a value greater than 0.</td>
</tr>
<tr>
<td>Tuning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum number of</td>
<td>-nMemParts</td>
<td>Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the Hub. The utility that consumes the Derive Data from Hub (mpxredvd) output (such as Generate Frequency Stats (mpxfreq)) must use a matching “memparts” value. A good rule of thumb is to leave this at the default unless you need the memory. The higher the member partitions the slower your mpxcomp process, as the Hub must do more duplicate comparisons. Default: 1</td>
</tr>
<tr>
<td>Member partitions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum number of</td>
<td>-nBktParts</td>
<td>Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the Hub. A good rule of thumb is to leave this at the default unless you need the memory. Default: 10</td>
</tr>
<tr>
<td>Bucket partitions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum number of</td>
<td>-nQryParts</td>
<td>Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the Hub. A good rule of thumb is to leave this at the default unless you need the memory. This option is enabled only when the option to Generate query BXM is also enabled. Default: 1</td>
</tr>
<tr>
<td>Query partitions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block size</td>
<td>-blkSize</td>
<td>Number of members in a block. Default: 1000</td>
</tr>
<tr>
<td>Buffer size</td>
<td>-buffSize</td>
<td>Size for each file I/O buffer. Default: 65536</td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encoding</td>
<td>-encoding</td>
<td>Choices are Latin1, UTF8 and UTF16. Default: latin1</td>
</tr>
<tr>
<td>Minimum bucket role</td>
<td>-minBktTag</td>
<td>The lowest bucketing role designation used in the algorithm to include in the process. Default: 0 (use all)</td>
</tr>
<tr>
<td>Maximum bucket role</td>
<td>-maxBktTag</td>
<td>The highest bucketing role designation used in the algorithm to include in the process. Default: 0 (use all)</td>
</tr>
<tr>
<td>Minimum query role</td>
<td>-minQryRole</td>
<td>The lowest query role designation used in the algorithm to include in the process. This option is enabled only when the option to Generate query BXM is also enabled. Default: 10000</td>
</tr>
</tbody>
</table>
Table 9. Derive Data from Hub (mpxredvd) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum member record number</td>
<td>-minMemRecno</td>
<td>Specifies the lowest MEMRECNO to include in the process. Default: 0 (use all)</td>
</tr>
<tr>
<td>Maximum member record number</td>
<td>-maxMemRecno</td>
<td>Specifies the highest MEMRECNO to include in the process. Default: 0 (use all)</td>
</tr>
</tbody>
</table>

Log Options

- Trace logging: Produces a trace of activity as interactions flow through the system. This option is very verbose and should only be used for short periods of time.
- Debug logging: Produces low-level diagnostics used internally by Initiate Systems to identify what was happening on the system before an error condition occurred. This option generates a large amount of output per activity and should only be used for short periods of time. **Note:** Debug logging can potentially include personal member information such as member identification number, name, etc.
- Timer logging: Produces timings on certain operations to help identify where significant processing time is elapsing.
- SQL logging: Outputs the SQL that is sent by the Master Data Engine database layer to the RDBMS. This helps in diagnosing database-related issues. This option can produce large amounts of output depending on the activity.
- Audit logging: Produces activity information and non-critical warnings. Often, this option is used when a new system is first implemented to monitor activity.
- Algorithm logging: A separate logging level for algorithm-related debug information without the risk of including protected health information (PHI). Default: disabled

Derive Data from UNLs (mpxfsdvd)

This utility is a data derivation method that uses pre-existing member unload files to extract and create comparison strings, bucket hashes, and binaries. It is most commonly used when you have made changes to your algorithm, but the data itself has not changed.

Table 10. Derive Data from UNLs (mpxfsdvd) options

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Type</td>
<td>-memType</td>
<td>If you have multiple member types in the Hub database and only need to derive data for one of those member types, select the desired member type here; otherwise, select ALL. Default: ALL</td>
</tr>
</tbody>
</table>
Table 10. Derive Data from UNLs (mpxfsdvd) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
</table>
| UNL input  | -unlInpDir          | Derive Data from UNLs (mpxfsdvd) reads the member attribute data (specified on the Attribute Types tab) from the unl files in the directory specified here. This directory is relative to the project's work directory on the Hub:  
  \(hub\_instance\_path\}\inst\mpinet\_hub\_instance\_name\work\project\_name\work\UNL\_input\_dir\)  
  Default: unl                                                                                                          |
| Generate UNL output | -unlOutDir and -unlOutSegs used together | Indicates whether Derive Data from UNLs (mpxfsdvd) should generate UNL files during processing. Also with -unlOutSegs, instructs mpxfsdvd to create UNL files containing bucketing data or comparison data, or instructs mpxfsdvd to generate query UNL files during processing (the files are used by the relationship linker).  
  Default: enabled                                                                                                      |
| UNL output directory | -unlOutDir          | The output of Derive Data from UNLs (mpxfsdvd) is the derived data segments (comparison, bucket and, optionally, query data), which have their own unl files (mpi_memcmpd, mpi_membktd, and mpi_memqryd, respectively) written to the directory specified here. This directory is relative to the project's work directory on the Hub:  
  \(hub\_instance\_path\}\inst\mpinet\_hub\_instance\_name\work\project\_name\work\UNL\_output\_dir\)  
  Default: unl                                                                                                          |
| Generate Bucket UNL | use -unlOutSegs with -unlOutDir | Instructs Derive Data from UNLs (mpxfsdvd) to create UNL files containing bucketing data.  
  Default: enabled                                                                                                      |
| Generate Comparison UNL | use -unlOutSegs with -unlOutDir | Instructs Derive Data from UNLs (mpxfsdvd) to create UNL files containing comparison data.  
  Default: enabled                                                                                                      |
| Generate Query UNL | use -unlOutSegs with -unlOutDir | Instructs Derive Data from UNLs (mpxfsdvd) to generate query UNL files during processing. These files are used by the relationship linker.  
  Default: disabled                                                                                                     |
| Generate BXM output | -{no}bxmCmpd and -{no}bxmBktd | Instructs Derive Data from UNLs (mpxfsdvd) to generate output files for bulk cross matching.  
  Default: enabled                                                                                                      |
| BXM output directory | -bxmOutDir          | Indicates where you want the .bxm output files to be located. This directory is relative to the project's work directory on the Hub:  
  \(hub\_instance\_path\}\inst\mpinet\_hub\_instance\_name\work\project\_name\work\BXM\_output\_dir\)  
  Default: bxm                                                                                                          |
Table 10. Derive Data from UNLs (mpxfsdvd) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate query BXM</td>
<td>-(no)bxmQryd</td>
<td>Indicates whether Derive Data from UNLs (mpxfsdvd) should generate query BXM files during processing. These files are used by the relationship linker.  Default: disabled</td>
</tr>
<tr>
<td>Generate SQL script for possible missing memheads</td>
<td>-(no)HeadSql</td>
<td>Generates SQL output. Instructs Derive Data from UNLs (mpxfsdvd) to generate an SQL file in the specified UNL output directory. If a UNL output directory is not specified then the output will be written to the BXM output directory. This SQL file will contain a query against the mpi_memhead table for members that were identified as missing. These members are identified when there is an attribute row that does not have a corresponding head row. Default: disabled</td>
</tr>
<tr>
<td>Performance Tuning</td>
<td></td>
<td>These fields act as a filter to include buckets up to the maximum (maximum bucket role), above a minimum (minimum bucket role) or within a range if both are set to a value greater than 0.</td>
</tr>
<tr>
<td>Maximum number of Member partitions</td>
<td>-nMemParts</td>
<td>Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the Hub. The utility that consumes the Derive Data from UNLs (mpxfsdvd) output (such as Generate Frequency Stats (mpxfreq)) must use a matching “memparts” value. A good rule of thumb is to leave this at the default unless you need the memory. The higher the member partitions the slower your mpxcomp process, as the Hub must do more duplicate comparisons. Default: 1</td>
</tr>
<tr>
<td>Maximum number of Bucket partitions</td>
<td>-nBktParts</td>
<td>Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the Hub. A good rule of thumb is to leave this at the default unless you need the memory. Default: 10</td>
</tr>
<tr>
<td>Maximum number of Query partitions</td>
<td>-nQryParts</td>
<td>Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the Hub. A good rule of thumb is to leave this at the default unless you need the memory. This option is enabled only when the option to Generate query BXM is also enabled. Default: 1</td>
</tr>
<tr>
<td>Buffer size</td>
<td>-buffSize</td>
<td>Size for each file I/O buffer. Default: 65536</td>
</tr>
</tbody>
</table>

Options

| Encoding                                      | -encoding          | Choices are Latin1, UTF8 and UTF16. Default: latin1                                                                                                                                                   |

Chapter 3. Jobs and job sets  61
<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum bucket role</td>
<td>-minBktTag</td>
<td>The lowest bucketing role designation used in the algorithm to include in the process. Default: 0</td>
</tr>
<tr>
<td>Maximum bucket role</td>
<td>-maxBktTag</td>
<td>The highest bucketing role designation used in the algorithm to include in the process. Default: 0</td>
</tr>
<tr>
<td>Minimum query role</td>
<td>-minQryRole</td>
<td>The lowest query role designation used in the algorithm to include in the process. This option is enabled only when the option to Generate query BXM is also enabled. Default: 10000</td>
</tr>
<tr>
<td>Maximum errors before stopping</td>
<td>-maxErrs</td>
<td>Maximum errors before halting processing. This option sets a threshold for errors in the data. Once the threshold is reached, Derive Data from UNLs (mpxfsdvd) stops processing. The intent of this option is to allow the user to process a set of input UNLs with tolerance for data issues. For example, if the UNL has an incorrect number of fields, the member record will be rejected and re-derivation will not complete for that member. The mpxfsdvd utility writes detailed information into the log file including the line number, input file and reason for the rejection. Default: 100</td>
</tr>
<tr>
<td>Number of records to skip</td>
<td>-skipRecs</td>
<td>Number of member records to skip before re-deriving members from the specified input files. Processing will begin with the next member read from MEMHEAD after skipping this number of records. Default: 0</td>
</tr>
<tr>
<td>Maximum number of records to process</td>
<td>-maxRecs</td>
<td>Maximum number of member records to re-derive from the specified input files. When using this parameter along with skipping member records, this number will include the number skipped. Default: Process all records</td>
</tr>
<tr>
<td>Enable incremental cross match</td>
<td>-ixmMode</td>
<td>Use this option to enable incremental cross matching. In IXM mode, a subset of members are compared rather than the entire member set. If running a BXM, use the default of false. If running an IXM, set this to true. Default: disabled</td>
</tr>
<tr>
<td>Attribute Types</td>
<td></td>
<td>Here you can select the attribute types that Derive Data from UNLs (mpxfsdvd) will read from the input UNL files.</td>
</tr>
<tr>
<td>Log Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trace logging</td>
<td></td>
<td>Produces a trace of activity as interactions flow through the system. This option is very verbose and should only be used for short periods of time.</td>
</tr>
</tbody>
</table>
Table 10. Derive Data from UNLs (mpxsdvd) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug logging</td>
<td></td>
<td>Produces low-level diagnostics used internally by Initiate Systems to identify what was happening on the system before an error condition occurred. This option generates a large amount of output per activity and should only be used for short periods of time. <strong>Note:</strong> Debug logging can potentially include personal member information such as member identification number, name, etc.</td>
</tr>
<tr>
<td>Timer logging</td>
<td></td>
<td>Produces timings on certain operations to help identify where significant processing time is elapsing.</td>
</tr>
<tr>
<td>SQL logging</td>
<td></td>
<td>Outputs the SQL that is sent by the Master Data Engine database layer to the RDBMS. This helps in diagnosing database-related issues. This option can produce large amounts of output depending on the activity.</td>
</tr>
<tr>
<td>Audit logging</td>
<td></td>
<td>Produces activity information and non-critical warnings. Often, this option is used when a new system is first implemented to monitor activity.</td>
</tr>
<tr>
<td>Algorithm logging</td>
<td></td>
<td>A separate logging level for algorithm-related debug information without the risk of including protected health information (PHI).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: disabled</td>
</tr>
</tbody>
</table>

Export Linkage History (mpxxeia)

This cross match program is used to unload existing entity linkage data from the database (unloads the mpi_entxeia_<enttype> table to binary format). If the Link Entities (mpxlink) utility will be run with the “Force xeia information to default to existing information” option enabled, run the Export Linkage History (mpxxeia) utility first.

Table 11. Export Linkage History (mpxxeia) options

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type</td>
<td>-entType</td>
<td>This option identifies the type of entity being computed. If you are implementing multiple entity types (e.g., identity and household), you will have to run Link Entities (mpxlink) for each type. This option is required and there is no default setting.</td>
</tr>
</tbody>
</table>
### Table 11. Export Linkage History (mpxxeia) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
</table>
| BXM output directory | -bxmOutDir    | The directory in which you want the Export Linkage History (mpxxeia) output binary files located. Binary output files are used by the relationship linkers. This directory is typically relative to the work directory on the server hosting the Hub, such as:  
  
  \`hub_instance_path\inst\mpinet_hub_instance_name\work\projectname\work\ BXM_output_dir\`  
  
  Generating output in binary form is optional; specifying an output directory with this option is what causes binary output to be generated. In other words, if no directory is specified here, no binary output is generated.  
  
  Default: bxm                                                                                                                                  |

#### Log Options

<table>
<thead>
<tr>
<th>Log Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace logging</td>
<td>Produces a trace of activity as interactions flow through the system. This option is very verbose and should only be used for short periods of time.</td>
</tr>
</tbody>
</table>
| Debug logging | Produces low-level diagnostics used internally by Initiate Systems to identify what was happening on the system before an error condition occurred. This option generates a large amount of output per activity and should only be used for short periods of time.  
  
  **Note:** Debug logging can potentially include personal member information such as member identification number, name, etc.                                                      |
| Timer logging | Produces timings on certain operations to help identify where significant processing time is elapsing.                                                                                                     |
| SQL logging  | Outputs the SQL that is sent by the Master Data Engine database layer to the RDBMS. This helps in diagnosing database-related issues. This option can produce large amounts of output depending on the activity. |
| Audit logging | Produces activity information and non-critical warnings. Often, this option is used when a new system is first implemented to monitor activity.                                                                |
| Algorithm logging | A separate logging level for algorithm-related debug information without the risk of including protected health information (PHI).  
  
  Default: disabled                                                                                                                                   |

### Link Entities (mpxlink)

Link Entities (mpxlink) is the cross match program, which, along with mpxcomp, makes up the Bulk Cross-Match (BXM) operation and enables entity linkage. Link Entities (mpxlink) takes comparison results from the mpxcomp or Sort Binary Link Data (mpxsort) operation and creates entity link and task files (UNL files) that can be loaded into the database.
<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type</td>
<td>-entType</td>
<td>This option identifies the type of entity being computed. If you are implementing multiple entity types (e.g., identity and household), you will have to run Link Entities (mpxlink) for each type. This option is required and there is no default setting.</td>
</tr>
<tr>
<td>Inputs and Outputs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BXM input directory</td>
<td>-bxmInpDir</td>
<td>The directory where the input binary (.bin) files to link are stored. Input files may be from mpxcomp, or other processes such as an IXM (incremental cross match). This directory is typically the work directory on the server hosting your Hub configuration. This option is required and there is no default setting. You can list multiple directories for this option; separate multiple directories with single spaces. Default: bxm</td>
</tr>
<tr>
<td>UNL output directory</td>
<td>-unlOutDir</td>
<td>The directory in which you want the Link Entities (mpxlink) output binary files located. Binary output files are used by the relationship linkers. The binary output file is named mpx_bxmxmem.bin. This directory is typically relative to the work directory on the server hosting the Hub configuration. Generating output in binary form is optional; specifying an output directory with this option is what causes binary output to be generated. In other words, if no directory is specified here, no binary output is generated. Default: unl</td>
</tr>
<tr>
<td>BXM output directory</td>
<td>-bxmOutDir</td>
<td>Indicates where you want the bxm output files to be located. This directory is relative to the project's work directory on the Hub: hub_instance_path\inst\mpinet_hub_instance_name\work\projectname\work\BXM_output_dir Default: bxm</td>
</tr>
<tr>
<td>Write mpi_entlink.unl</td>
<td>-entLink</td>
<td>Instructs the Hub to write new linkages and entity level tasks to a UNL file (mpi_entlink.unl). Default: enabled</td>
</tr>
<tr>
<td>Write mpi_entxeia.unl</td>
<td>-entXeia</td>
<td>Instructs the Hub to write historical Enterprise ID data to a UNL file (mpi_entxeia.unl). Default: enabled</td>
</tr>
<tr>
<td>Write mpi_entxtsk.unl</td>
<td>-entXtsk</td>
<td>Instructs the Hub to write information about tasks related to an entity to a UNL file (mpi_entxtsk.unl). Default: enabled</td>
</tr>
<tr>
<td>Workbench</td>
<td>Command line</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td><strong>Number of member partitions</strong></td>
</tr>
<tr>
<td>Tuning</td>
<td>-nMemParts</td>
<td>Member partitions (MemParts) are used to partition up the data set. Typically this is done for memory considerations: since Link Entities (mpxlink) requires the entire input data set (e.g., the binary files of comparison results) to be read into memory at once, breaking the data set into smaller pieces allows them to fit into the available memory on the system. MemParts differs from MxmParts (described below) in that MemParts breaks up the memHead and memCmpd data files, whereas MxmParts breaks up link and task files (the output of the mpxcomp utility). The MemParts value set here must be the same as the MemParts value set in mpxcomp, and in the utility that created the input for mpxcomp (mpxfsdvd, mpxprep, mpxredvd, etc). In other words, the MemParts setting in mpxcomp determines how many partitioned file segments are passed to Link Entities (mpxlink); the Link Entities (mpxlink) MemParts setting must accurately reflect the number of partitioned file segments coming from mpxcomp. There is a performance consideration to partitioning the data set: the higher the MemParts is set, the slower the Link Entities (mpxlink) process will be. Default: 1 Leave this value set to 1 unless memory is an issue. The maximum value is 100.</td>
</tr>
</tbody>
</table>
Table 12. Link Entities (mpxlink) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of output partitions</td>
<td>-nMxmParts</td>
<td>Like MemParts, the MxmParts option partitions the output of the mpxcomp process; as with MemParts, this option is used when the output file is too large to be read into memory in its entirety, and needs to be broken up into smaller sections in order to fit into available memory. MxmParts differs from MemParts (described above) in that MxmParts breaks up link and task files (the output of the mpxcomp utility), whereas MemParts breaks up the memHead and memCmpd data files. The MxmParts value set here must be the same as the MxmParts value set in mpxcomp, which provides the input to Link Entities (mpxlink). In other words, the MxmParts setting in mpxcomp or Sort Binary Link Data (mpxsort) determines how many partitioned file segments are passed to mpxlink; the mpxlink MxmParts setting must accurately reflect the number of partitioned file segments coming from mpxcomp or Sort Binary Link Data (mpxsort). Default: 1 Leave this value set to 1 unless memory is an issue. The maximum value is 100.</td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use member rule records from mpxprep</td>
<td>-{no}bxmRule</td>
<td>Use member rule records from mpxprep, mpxredvd, and/or mpxfsdvd. Member rules express the relationship between the survivor and obsolete members in a merge. The input data used here is created by default in Prepare Binary Files (mpxprep); it is therefore not necessary to specify a corresponding option in mpxprep. Default: enabled</td>
</tr>
<tr>
<td>Use linkage records from mpxcomp</td>
<td>-{no}bxmLink</td>
<td>Link Entities (mpxlink) uses this data to form entities that can be loaded into the database. The input data used here is created with the corresponding Use linkage records from mpxcomp (-bxmLink) option in mpxcomp. Default: enabled</td>
</tr>
<tr>
<td>Use task records from mpxcomp</td>
<td>-{no}bxmTask</td>
<td>Link Entities (mpxlink) uses this data to form tasks that can be loaded into the database. The input data used here is created with the corresponding Use task records from mpxcomp (-bxmTask) option in mpxcomp. Default: enabled</td>
</tr>
<tr>
<td>Workbench</td>
<td>Command line</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Use reviewid records from mpxcomp</td>
<td>-{no}bxmRvid</td>
<td>Link Entities (mpxlink) uses this data to form review identifier tasks that can be loaded into the database. The input data used here is created with the corresponding <code>Use reviewid records from mpxcomp (-bxmRvid)</code> option in mpxcomp. Default: enabled</td>
</tr>
<tr>
<td>Use potential duplicate task records from entxtsk</td>
<td>-{no}bxmPD</td>
<td>Use PD (potential duplicate) task records from the mpxxtask utility (entxtsk). Link Entities (mpxlink) uses this data to form review identifier tasks that can be loaded into the database. Default: disabled</td>
</tr>
<tr>
<td>Use potential linkage task records from entxtsk</td>
<td>-{no}bxmPL</td>
<td>Use PL (potential linkage) task records from the mpxxtask utility (entxtsk), which captures existing task information from the database. Default: disabled</td>
</tr>
<tr>
<td>Use review identifier task records from entxtsk</td>
<td>-{no}bxmRI</td>
<td>Use RI (review identifier) task records from the mpxxtask utility (entxtsk), which captures existing task information from the database. Default: disabled</td>
</tr>
<tr>
<td>Use explicit different records from entrule</td>
<td>-{no}bxmDiff</td>
<td>This option controls whether Link Entities (mpxlink) uses existing entity rules when forming entities. For example, if the users broke two members in an entity apart in Inspector, a non-identity rule is created by the engine (likewise if two members are manually linked, an &quot;identity&quot; rule is created). Mpxrule captures these rules as &quot;same&quot; or &quot;diff&quot; rules (identity or non-identity, respectively). If the user is &quot;re-crossmatching&quot; an existing database, including these rules will prevent Link Entities (mpxlink) from reforming linkages (in the case of diff rules), or force members to be in the same entity (in the case of a &quot;same&quot; rule). The input data used here is created with the corresponding <code>Use explicit different records from entrule (-bxmDiff)</code> option in mpxcomp. If this option is enabled, mpxrule must be run with the same option before running mpxlink. Default: disabled</td>
</tr>
<tr>
<td>Use explicit same records from entrule</td>
<td>-{no}bxmSame</td>
<td>Like -bxmDiff, this option controls whether Link Entities (mpxlink) uses existing entity rules when forming entities. Refer to the description for -bxmDiff, above, for more details. The input data used here is created with the corresponding <code>Use explicit same records from entrule (-bxmSame)</code> option in mpxcomp. If this option is enabled, mpxrule must be run with the same option before running mpxlink. Default: disabled</td>
</tr>
</tbody>
</table>
Table 12. Link Entities (mpxlink) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use implicit link records from entlink</td>
<td>-{no}bxmXeia</td>
<td>This option instructs Link Entities (mpxlink) to include the output from the Export Linkage History (mpxxeia) utility. Export Linkage History (mpxxeia) captures existing entity data. The input data used here is created with the corresponding Use implicit link records from entlink (-bxmXeia) option in mpxcomp. Default: disabled</td>
</tr>
<tr>
<td>Enable incremental cross match</td>
<td>-ixmMode</td>
<td>Use this option to enable incremental cross matching. In IXM mode, a subset of members are compared rather than the entire member set. If running a BXM, use the default of false. If running an IXM, set this to true. Default: disabled</td>
</tr>
<tr>
<td>Compute full tskset information</td>
<td>-{no}tskSets</td>
<td>Assigns a task set number to a member in a task. A task set identifies a group (two or more) of records explicitly identified as being in a task. For example, if memrecnos 1, 2, 3 are in a Potential Duplicate task, they are all assigned tskset=1; memrecno 4 &amp; 5 are in a Potential Linkage task, they are assigned tskset=2, etc.). This data is typically used for reporting purposes. Default: disabled</td>
</tr>
<tr>
<td>Count task related members</td>
<td>-{no}tskRelatedMembers</td>
<td>When enabled, this option creates a count of members in a task so that when you have a trigger member, you can tell that there are ( n ) members in the task. The count is only calculated when a member is cross matched. Default: enabled</td>
</tr>
<tr>
<td>Force xeia information to default to existing information</td>
<td>-{no}strict</td>
<td>Use this option to force xeia (entity linkage) data to default to existing information (rules and prior data). Setting this option to ( \text{strict} ) makes Link Entities (mpxlink) sensitive to anomalies in the data. Disable this option to instruct Link Entities (mpxlink) to ignore anomalies in the data, such as inconsistencies or discrepancies arising from live updates to the table (that is, discrepancies that might occur because data is changing via updates as it is being collected by the mpx utilities that create the input for Link Entities (mpxlink)). This option is typically used for reporting purposes. Default: enabled</td>
</tr>
<tr>
<td>Starting entrecno for .unl</td>
<td>-entRecno</td>
<td>Enables you to specify a starting entity record number to create the mpi_entlink_xx.unl file. The parameter is optional; if not set, Link Entities (mpxlink) defaults to applying 1 as the starting entity record number. Default: 1</td>
</tr>
<tr>
<td>Workbench</td>
<td>Command line</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Generate entBktd information for non-transitive entities</td>
<td>-entBktd</td>
<td>This option and the <strong>Bucket output directory</strong> (used together) allow Link Entities (mpxlink) to generate a binary bucket file that can be consumed by mpxcomp to rescore members that exist in the same transitive entity. This is leveraged for non-transitive entities to get scores between members who were brought together by a “glue” member and would not have a score generated by our traditional binary bucket file generated during Prepare Binary Files (mpxprep) or Derive Data from UNLs (mpxfsdvd). This allows a second pass using mpxcomp and Link Entities (mpxlink) to produce more accurate non-transitive entities. Although non-transitive entities can be produced using a single pass through mpxcomp and Link Entities (mpxlink), the two-pass approach improves accuracy. Default: disabled</td>
</tr>
<tr>
<td>Bucket output directory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auditing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write mpi_audhead.unl</td>
<td>-{no}audHead</td>
<td>This option instructs Link Entities (mpxlink) to write an mpi_audhead.unl file, and uses the audrecno specified in the -audRecno (Common audit record number for all .unl) option. This option is commonly used in new implementations where no audit records exist yet. Default: disabled</td>
</tr>
<tr>
<td>Common audit record number for all .unl</td>
<td>-audRecno</td>
<td>This option sets the audit record number for the .unl files that will be loaded into the mpi_audhead database table. When the <strong>Write mpi_audhead.unl</strong> option is enabled, you can set this to an existing mpi_audhead record number. Default: 2 Maximum: 100</td>
</tr>
<tr>
<td>Common user record number for all .unl</td>
<td>-usrRecno</td>
<td>This option sets the user record number for the .unl files that will be loaded into the mpi_audhead database table. When the <strong>Write mpi_audhead.unl</strong> option is enabled, you can set this to an existing mpi_usrhead user record number. Default: 1 Maximum: 100</td>
</tr>
<tr>
<td>Transaction record number for audit record</td>
<td>-ixnRecno</td>
<td>This option sets the transaction record number for the .unl files that will be loaded into the mpi_audhead database table. When the <strong>Write mpi_audhead.unl</strong> option is enabled, you can set this to an existing mpi_ixnhead user record number. Default: 71 Maximum: 100</td>
</tr>
</tbody>
</table>
Table 12. Link Entities (mpxlink) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event type number for audit record</td>
<td>-evtTypeno</td>
<td>Use this option to specify an event type for the audhead records. When the Write mpi_audhead.unl option is enabled, you can set this to an existing mpi_evttype event type number. Event types are defined in &quot;Events&quot; on page 120. Default: 0 Maximum: 100</td>
</tr>
</tbody>
</table>

Log Options

| Trace logging | Produces a trace of activity as interactions flow through the system. This option is very verbose and should only be used for short periods of time. Default: disabled |
| Debug logging | Produces low-level diagnostics used internally by Initiate Systems to identify what was happening on the system before an error condition occurred. This option generates a large amount of output per activity and should only be used for short periods of time. Note: Debug logging can potentially include personal member information such as member identification number, name, etc. Default: disabled |
| Timer logging | Produces timings on certain operations to help identify where significant processing time is elapsing. Default: disabled |
| SQL logging | Outputs the SQL that is sent by the Master Data Engine database layer to the RDBMS. This helps in diagnosing database-related issues. This option can produce large amounts of output depending on the activity. Default: disabled |
| Audit logging | Produces activity information and non-critical warnings. Often, this option is used when a new system is first implemented to monitor activity. Default: disabled |
| Algorithm logging | A separate logging level for algorithm-related debug information without the risk of including protected health information (PHI). Default: disabled |

Link Relationships

Creates relationship linkages in bulk fashion. The steps required prior to running the Link Relationships depend on whether you have added relationship type rules or made changes to the algorithm. The relationship linker is run on the command line using the MADCONFIG madeng_rel_linker target.
### Table 13. Steps for executing Link Relationships

<table>
<thead>
<tr>
<th>Step</th>
<th>No relationship type rules were added and algorithm was not changed</th>
<th>Relationship type rules were added and/or the algorithm was changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Run the Prepare Binary Files (mpxprep) job on all entities that have relationships defined, with the Generate Query BXM option enabled. The BXM output directory for Prepare Binary Files (mpxprep) and Link Entities (mpxlink) (step 3 below) must match. Refer to &quot;Prepare Binary Files (mpxprep)&quot; on page 73.</td>
<td>Run the Derive Data from Hub (mpxredvd) job on all entities that have relationships defined, with the Generate Query BXM option enabled. The BXM output directory for Derive Data from Hub (mpxredvd) (or Derive Data from UNLs (mpxfsdvd)) and Link Entities (mpxlink) (step 3 below) must match. Refer to &quot;Derive Data from Hub (mpxredvd)&quot; on page 56. (Alternatively, you can execute mpxfsdvd instead, with the Generate Query BXM option enabled, if UNL files are available. Refer to &quot;Derive Data from UNLs (mpxfsdvd)&quot; on page 59.)</td>
</tr>
<tr>
<td>2</td>
<td>Run the Compare Members in Bulk (mpxcomp) job on all entities that have relationships defined. Refer to &quot;Compare Members in Bulk (mpxcomp)&quot; on page 40.</td>
<td>Run the Compare Members in Bulk (mpxcomp) job on all entities that have relationships defined. Refer to &quot;Compare Members in Bulk (mpxcomp)&quot; on page 40.</td>
</tr>
<tr>
<td>3</td>
<td>Run the Link Entities (mpxlink) job on all entities that have relationships defined, with the Generate BXM Output option enabled. The BXM output directory specified must match what was used for Prepare Binary Files (mpxprep), Derive Data from UNLs (mpxfsdvd) or Derive Data from Hub (mpxredvd) in step 1. Refer to &quot;Link Entities (mpxlink)&quot; on page 64.</td>
<td>Run the Link Entities (mpxlink) job on all entities that have relationships defined, with the Generate BXM Output option enabled. The BXM output directory specified must match what was used for Prepare Binary Files (mpxprep), Derive Data from UNLs (mpxfsdvd) or Derive Data from Hub (mpxredvd) in step 1. Refer to &quot;Link Entities (mpxlink)&quot; on page 64.</td>
</tr>
<tr>
<td>4</td>
<td>Run the Link Relationships job on all entities that have relationships defined.</td>
<td>Run the Link Relationships job on all entities that have relationships defined.</td>
</tr>
</tbody>
</table>

### Table 14. Link Relationships options

<table>
<thead>
<tr>
<th>Workbench options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inputs and Outputs</strong></td>
<td></td>
</tr>
<tr>
<td>Input directory</td>
<td>Specifies the location on the Hub where the bulk cross-match results are located.</td>
</tr>
<tr>
<td>Default: bxm</td>
<td></td>
</tr>
<tr>
<td>Output directory</td>
<td>Specifies the location on the Hub where the relationship linkage files will be saved.</td>
</tr>
<tr>
<td>Default: unl</td>
<td></td>
</tr>
</tbody>
</table>
### Table 14. Link Relationships options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relationship type</td>
<td>Indicates the relationship type for the job. You can specify a single relationship type or all relationship types.</td>
</tr>
<tr>
<td></td>
<td>Default: ALL</td>
</tr>
</tbody>
</table>

#### Performance Tuning

| Maximum JVM heap size          | Indicates the maximum heap size set in the java virtual machine. This is set in the JVM using a command such as java -Xms64m -Xmx512m                                                                                                     |
|                                | Default: 256                                                                                                                                                                                               |

<table>
<thead>
<tr>
<th>Number of parts for map/set data structures</th>
<th>Indicates the number of parts into which the Link Relationships will divide the data for processing. Estimate the number of parts based on the number of expected links divided by 120 million. Failure to set the number of parts high enough may cause the Link Relationships to fail. However, setting it too high may cause it to run slower than is optimal.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Default: 3</td>
</tr>
</tbody>
</table>

### Prepare Binary Files (mpxprep)

This cross match program is used to generate Bulk Cross-Match (BXM) data.

**Table 15. Prepare Binary Files (mpxprep) options**

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member type</td>
<td>-memType</td>
<td>If you have multiple member types in the Hub database and only need to generate BXM data for one of those member types, the Member Type filter may be used. All entity types for that member are processed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: All member types</td>
</tr>
</tbody>
</table>

| Inputs and Outputs | Output directory | -bxmOutDir | Where any .bin output files will be located. This directory is relative to the instance Work Directory on the server hosting the Hub.                                                                                     |
|                   |                  |           | Default: bxm                                                                                                                                                                                                |

| Generate Query BXM | -{no}bxmQryd | This option is for use with the relationship linker and instructs Prepare Binary Files (mpxprep) to create BXM files containing query data. The relationship types, attributes and rules should already be defined so that mpxprep knows what data to include in the BXM file. |
|                   |              |           | Default: disabled                                                                                                                                                                                             |
### Table 15. Prepare Binary Files (mpxprep) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of Member Partitions</td>
<td>-nMemParts</td>
<td>Member partitions (MemParts) are used to partition up the data set. Typically this is done for memory considerations: since Link Entities (mpxlink) requires the entire input data set (e.g., the binary files of comparison results) to be read into memory at once, breaking the data set into smaller pieces allows them to fit into the available memory on the system. MemParts breaks up the memHead and memCmpd data files. If you set a value other than 1 here, you must set a matching MemParts value for any “downstream” utility that uses the output of Prepare Binary Files (mpxprep) (such as mpxcomp or mpxlink). In other words, the MemParts setting in those downstream utilities must accurately reflect the number of partitioned file segments coming from Prepare Binary Files (mpxprep). Leave this value set to 1 unless memory is an issue. Default: 1 Maximum: 100</td>
</tr>
<tr>
<td>Maximum number of Bucket Partitions</td>
<td>-nBktParts</td>
<td>Like Maximum number of Member Partitions, the Maximum number of Bucket Partitions option partitions the output of the Prepare Binary Files (mpxprep) process; as with MemParts, this option is used when the output file is too large to be read into memory in its entirety, and needs to be broken up into smaller sections in order to fit into available memory. BktParts differs from MemParts in that it breaks up the membkt data. This is the most common option for reducing your memory footprint (and can also help sort performance on large data sets). Default: 1 Maximum: 100</td>
</tr>
<tr>
<td>Maximum number of Query partitions</td>
<td>-nQryParts</td>
<td>Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the Hub. A good rule of thumb is to leave this at the default unless you need the memory. This option is enabled only when the option to Generate query BXM is also enabled. Default: 1 Maximum: 100</td>
</tr>
<tr>
<td>Block Size (number of members)</td>
<td>-blkSize</td>
<td>Number of members Default: 1000</td>
</tr>
<tr>
<td>File IO buffer size</td>
<td>-buffSize</td>
<td>Size (in bytes) for each file IO buffer. Default: 65536</td>
</tr>
</tbody>
</table>

Options
<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Bucket Role</td>
<td>-minBktTag</td>
<td>This specifies the lowest bucketing role to be included in the operation. Default: 0</td>
</tr>
<tr>
<td>Maximum Bucket Role</td>
<td>-maxBktTag</td>
<td>This specifies the highest bucketing role to be included in the operation. Default: 0 (include all)</td>
</tr>
<tr>
<td>Minimum Query role</td>
<td>-minQryRole</td>
<td>The lowest query role designation used in the algorithm to include in the process. This option is enabled only when the option to Generate query BXM is also enabled. Default: 10000</td>
</tr>
<tr>
<td>Minimum MemRecNo</td>
<td>-minMemRecno</td>
<td>Specifies the lowest MEMRECNO to include in the process. Default: 0 (use all records)</td>
</tr>
<tr>
<td>Maximum MemRecNo</td>
<td>-maxMemRecno</td>
<td>Specifies the highest MEMRECNO to include in the process. Default: 0 (use all records)</td>
</tr>
<tr>
<td>Log Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trace logging</td>
<td></td>
<td>Produces a trace of activity as interactions flow through the system. This option is very verbose and should only be used for short periods of time. Default: disabled</td>
</tr>
<tr>
<td>Debug logging</td>
<td></td>
<td>Produces low-level diagnostics used internally by Initiate Systems to identify what was happening on the system before an error condition occurred. This option generates a large amount of output per activity and should only be used for short periods of time. <strong>Note:</strong> Debug logging can potentially include personal member information such as member identification number, name, etc. Default: disabled</td>
</tr>
<tr>
<td>Timer logging</td>
<td></td>
<td>Produces timings on certain operations to help identify where significant processing time is elapsing. Default: disabled</td>
</tr>
<tr>
<td>SQL logging</td>
<td></td>
<td>Outputs the SQL that is sent by the Master Data Engine database layer to the RDBMS. This helps in diagnosing database-related issues. This option can produce large amounts of output depending on the activity. Default: disabled</td>
</tr>
<tr>
<td>Audit logging</td>
<td></td>
<td>Produces activity information and non-critical warnings. Often, this option is used when a new system is first implemented to monitor activity. Default: disabled</td>
</tr>
</tbody>
</table>
Table 15. Prepare Binary Files (mpxprep) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td></td>
<td>A separate logging level for algorithm-related debug information without</td>
</tr>
<tr>
<td>logging</td>
<td></td>
<td>the risk of including protected health information (PHI).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: disabled</td>
</tr>
</tbody>
</table>

### Sort Binary Link Data (mpxsort)

Sort Binary Link Data (mpxsort) reorders the bxmlink file when there are multiple memparts or multiple threads used during creation of the bxmlink file. This sort order is required by the non-transitive logic to keep the transitive entity sets grouped together so that members can be removed from the set (and possibly form additional entities) for the non-transitive phase.

Sort Binary Link Data (mpxsort) is run between the second Compare Members in Bulk (mpxcomp) and Link Entities (mpxlink) phase. The input to Sort Binary Link Data (mpxsort) is the output of mpxcomp. The **Maximum number of output partitions** option should match what was used for mpxcomp. Sort Binary Link Data (mpxsort) output is consumed by Link Entities (mpxlink). The options are all inline with current BXM/IXM options, with the exception of the **Use radix sort** option.

Table 16. Sort Binary Link Data (mpxsort) options

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type</td>
<td>-entType</td>
<td>This option identifies the type of entity being computed. If you are implementing multiple entity types (e.g., identity and household), you will have to run Sort Binary Link Data (mpxsort) for each type. This option is required and there is no default setting.</td>
</tr>
</tbody>
</table>

**Inputs and Outputs**

| Input directory | -bxmInpDir | This is the directory where the input binary (.bin) files from Derive Data and Create UNLs (mpxdata) are stored. This is typically the work directory created under hub_instance_path/inst/mpinet_hub_instance_name/work. Note that multiple Hubs might share the same hub_instance_path dir, but will have different hub_instance_names. The Hub instance that you connect to will dictate where the work directory lives. Default: bxm |

| Output directory | -bxmOutDir | This is the directory in which you want the Sort Binary Link Data (mpxsort) output binary files located. This is typically relative to the work directory on the server hosting the Hub configuration. Default: bxm |
### Table 16. Sort Binary Link Data (mpxsort) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of output partitions</td>
<td>-nMxmParts</td>
<td>This option partitions the output of Sort Binary Link Data (mpxsort) into smaller chunks for use by Link Entities (mpxlink). The value set for mpxcomp determines how many partitioned file segments are passed to Sort Binary Link Data (mpxsort) and Link Entities (mpxlink), thus the MxmParts value for all three must be the same.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maximum value: 100</td>
</tr>
<tr>
<td>Number of threads</td>
<td>-nthreads</td>
<td>The number of threads to used for the Sort Binary Link Data (mpxsort) process. The number of threads set can have an affect on system performance. The value should correspond to the number of CPUs available on the machine (e.g., if the machine has 4 CPUs, set the number of threads for mpxcomp to 4 to optimize the time it takes to process).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maximum value: 64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Recommended value: 1 thread per processor</td>
</tr>
</tbody>
</table>

### Options

- **Use radix sort** `-no.radixSort` Use radix sort instead of quick sort. A radix sort, also known as a bin sort, is an extremely fast method of sorting records. While a radix sort is orders of magnitude faster than a quick sort, the radix sort consumes twice as much memory as a quick sort does. On systems where memory is a constraint, you can disable radix sort and a quick sort will be used to save memory. On systems where memory is not an issue and maximum performance is required, radixsort can be used.

  Default: enabled

### Log Options

- **Trace logging** Produces a trace of activity as interactions flow through the system. This option is very verbose and should only be used for short periods of time.

- **Debug logging** Produces low-level diagnostics used internally by Initiate Systems to identify what was happening on the system before an error condition occurred. This option generates a large amount of output per activity and should only be used for short periods of time.

  **Note:** Debug logging can potentially include personal member information such as member identification number, name, etc.

- **Timer logging** Produces timings on certain operations to help identify where significant processing time is elapsing.

- **SQL logging** Outputs the SQL that is sent by the Master Data Engine database layer to the RDBMS. This helps in diagnosing database-related issues. This option can produce large amounts of output depending on the activity.
### Table 16. Sort Binary Link Data (mpxsort) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit logging</td>
<td></td>
<td>Produces activity information and non-critical warnings. Often, this option is used when a new system is first implemented to monitor activity.</td>
</tr>
<tr>
<td>Algorithm logging</td>
<td></td>
<td>A separate logging level for algorithm-related debug information without the risk of including protected health information (PHI).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: disabled</td>
</tr>
</tbody>
</table>

### Database Tools jobs

These jobs handle loading data into the Hub database and creating UNL files from data already in the Hub database.

**Load UNLs to DB (madunlload)**

This job reads data from .unl files and inserts it into the Hub database. This performs the same functionality as the command line utility madhubload.

### Table 17. Load UNLs to DB (madunlload) options

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load type</td>
<td>-objCode</td>
<td>The type of data to be loaded. Options are Member, Entity, Audit, Relationship, Tag, Dictionary, History.</td>
</tr>
<tr>
<td></td>
<td>(DIC, MEM,</td>
<td>Default: Member</td>
</tr>
<tr>
<td></td>
<td>AUD, REL, TAG, HST, ANL)</td>
<td></td>
</tr>
<tr>
<td>Member Type</td>
<td></td>
<td>This is enabled if the Load Type is Member and allows you to choose from the available member types in the current configuration.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: varies depending on available types</td>
</tr>
<tr>
<td>Entity Type</td>
<td></td>
<td>This is enabled if the Load Type is Entity and allows you to choose from the available entity types in the current configuration.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: varies depending on available types</td>
</tr>
<tr>
<td>UNL Input</td>
<td>-unldir</td>
<td>This path is relative to the project workspace directory on the Hub.</td>
</tr>
<tr>
<td>Directory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data to load</td>
<td>-tabList</td>
<td>The specific data elements to be loaded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 17. Load UNLs to DB (madunlload) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation type</td>
<td>-onepass</td>
<td>Select the database operation to perform:</td>
</tr>
<tr>
<td></td>
<td>-truncate</td>
<td><strong>Onepass</strong> performs the drop table, create table, load data, and create index operations. This is the best option when reloading large tables because it is fast and yields a clean index, which is not created until after the data is loaded.</td>
</tr>
<tr>
<td></td>
<td>-loaddata</td>
<td><strong>Truncate</strong> deletes all rows at once without removing the table or index. The table is then loaded with the index in place.</td>
</tr>
<tr>
<td></td>
<td>-index</td>
<td><strong>Loaddata</strong> performs the data load only (without rebuilding the table or index).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Index</strong> recreates the index. No data is loaded. It might be used after several loaddata operations if you had manually deleted the indexes to increase load speed.</td>
</tr>
<tr>
<td>Encoding</td>
<td>-encoding</td>
<td>Choices are latin1, UTF8 and UTF16. Default: latin1</td>
</tr>
<tr>
<td>Maximum errors before stopping</td>
<td>-maxErrs</td>
<td>This option sets a threshold for errors in the data. Once the threshold is reached, Load UNLs to DB (madunlload) stops (but that is not considered a madunlload error). The intent of this option is to allow you to process the operation with tolerance for known data issues. Default: 100</td>
</tr>
<tr>
<td>Don’t stop on errors</td>
<td></td>
<td>If checked, Load UNLs to DB (madunlload) will not halt processing when it encounters a data error. Checking this option disables the ‘Maximum errors before stopping’ option. Default: disabled</td>
</tr>
<tr>
<td>Number of records to process before commit</td>
<td>-commitSize</td>
<td>By default, Load UNLs to DB (madunlload) processes all records in the UNL file(s) before committing them to the database. If desired, specify the number of records to be committed at a time. Default: 0</td>
</tr>
<tr>
<td>Use internal database loader</td>
<td>-useint</td>
<td>By default, the job attempts to use a database vendor-supplied bulk loader. The internal loader uses ODBC insert calls to load the data one row at a time. This is useful when the RDBMS native loader is not available. The internal loader should be used in most cases to improve performance, except when loading an extremely large UNL file. If the internal loader is not used, the MAD_DBNAME environment variable needs to be configured for the engine process.</td>
</tr>
</tbody>
</table>
Table 17. Load UNLs to DB (madunlload) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The database is remote from the hub</td>
<td>-remote</td>
<td>Some databases need a special option passed to them when the database is not located on the same machine as the loading utility. If the database resides on a different computer than the Hub, enable this option.</td>
</tr>
<tr>
<td>Show SQL statements only (does not modify the database)</td>
<td>-noExec</td>
<td>To view the SQL statement that would be executed for this job without actually running the SQL against the Hub database.</td>
</tr>
<tr>
<td>Other database arguments</td>
<td>-otherArgs</td>
<td>Any additional arguments you want to be sent to the database.</td>
</tr>
<tr>
<td>Log Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trace logging</td>
<td></td>
<td>Produces a trace of activity as interactions flow through the system. This option is very verbose and should only be used for short periods of time. Default: disabled</td>
</tr>
</tbody>
</table>
| Debug logging                                                             |              | Produces low-level diagnostics used internally by Initiate Systems to identify what was happening on the system before an error condition occurred. This option generates a large amount of output per activity and should only be used for short periods of time.  
**Note:** Debug logging can potentially include personal member information such as member identification number, name, etc. Default: disabled                                                                                                                     |
| Timer logging                                                             |              | Produces timings on certain operations to help identify where significant processing time is elapsing. Default: disabled                                                                                                                                                                                                               |
| SQL logging                                                               |              | Outputs the SQL that is sent by the Master Data Engine database layer to the RDBMS. This helps in diagnosing database-related issues. This option can produce large amounts of output depending on the activity. Default: disabled                                                                                                                                      |
| Audit logging                                                             |              | Produces activity information and non-critical warnings. Often, this option is used when a new system is first implemented to monitor activity. Default: disabled                                                                                                                                                      |
| Algorithm logging                                                        |              | A separate logging level for algorithm-related debug information without the risk of including protected health information (PHI). Default: disabled                                                                                                                                               |

**Unload UNLS from DB (madunlunload)**

Creates .unl files from data in the Hub. The Hub database is left intact (data is not removed). This performs the same functionality as the command line utility madhubunload.
Table 18. Unload UNLS from DB (madunlunload) options

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unload type</td>
<td>-objCode</td>
<td>The type of data to be unloaded. Options are Member, Entity, Audit, Relationship, Tag, Dictionary. Default: Member</td>
</tr>
<tr>
<td></td>
<td>(DIC, MEM, AUD, REL, ANL)</td>
<td></td>
</tr>
<tr>
<td>Member Type</td>
<td></td>
<td>This is enabled if the Unload Type is Member, and allows you to choose from the available member types in the current configuration. Default: varies depending on available types</td>
</tr>
<tr>
<td>Entity Type</td>
<td></td>
<td>This is enabled if the Unload Type is Entity, and allows you to choose from the available entity types in the current configuration. Default: varies depending on available types</td>
</tr>
<tr>
<td>UNL Input Directory</td>
<td>-unlDir</td>
<td>This path is relative to the project workspace directory on the Hub. Default: unl</td>
</tr>
</tbody>
</table>

**Data to load**

The contents of this tab vary depending on the Load Type selected.

<table>
<thead>
<tr>
<th>Options</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoding</td>
<td>-encoding</td>
<td>Choices are Latin1, UTF8 and UTF16. Default: latin1</td>
</tr>
<tr>
<td>Number of records to process before commit</td>
<td>-commitSize</td>
<td>By default, Unload UNLS from DB (madunlunload) processes all records in the database before committing them to the UNL files. If desired, specify the number of records to be committed at a time. Default: 0</td>
</tr>
<tr>
<td>Show SQL statements only (does not modify the database)</td>
<td>-noExec</td>
<td>To view the SQL statement that would be executed for this job without actually running the SQL against the Hub database. Default: disabled</td>
</tr>
</tbody>
</table>

**Log Options**

| Trace logging | Produces a trace of activity as interactions flow through the system. This option is very verbose and should only be used for short periods of time. Default: disabled |
Table 18. Unload UNLS from DB (madununload) options (continued)

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
</table>
| Debug logging        |              | Produces low-level diagnostics used internally by Initiate Systems to identify what was happening on the system before an error condition occurred. This option generates a large amount of output per activity and should only be used for short periods of time.  
  **Note:** Debug logging can potentially include personal member information such as member identification number, name, etc.  
  Default: disabled                                                 |
| Timer logging        |              | Produces timings on certain operations to help identify where significant processing time is elapsing.  
  Default: disabled                                                   |
| SQL logging          |              | Outputs the SQL that is sent by the Master Data Engine database layer to the RDBMS. This helps in diagnosing database-related issues. This option can produce large amounts of output depending on the activity.  
  Default: disabled                                                   |
| Audit logging        |              | Produces activity information and non-critical warnings. Often, this option is used when a new system is first implemented to monitor activity.  
  Default: disabled                                                   |
| Algorithm logging    |              | A separate logging level for algorithm-related debug information without the risk of including protected health information (PHI).  
  Default: disabled                                                   |

**Full text index jobs**

This job category enables you to perform tasks such as indexing members.

**Index by Members job**

This job enables you to run the Index by Members utility from IBM Initiate Workbench, which indexes member data stored in the IBM Initiate Master Data Service.

This job updates all indexes in the cluster. It is not necessary to run the job on other members of a cluster. To execute this job from the Hub, save it in a job set, and then use the Run_JobSet utility on the command line.

Table 19. Index by Members options

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting member record identifier</td>
<td>Specify the member record number to start the range for the indexing process.</td>
</tr>
<tr>
<td>Ending member record identifier</td>
<td>Specify the member record number to end the range for the indexing process. The record number specified in this field is included.</td>
</tr>
</tbody>
</table>
Index by UNLs job
This job is to index member data from UNL files.

To execute this job from the Hub, save it in a job set, and then use the Run_JobSet utility on the command line.

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNL input directory</td>
<td>The directory on the Hub where the UNL files are located. This path is relative to the project workspace directory. You would typically use this job to index the UNL files produced by mpxdata, mpxprep, or other similar utility. The Index by UNLs job updates the index only on the node on which it is run. To update other nodes in a cluster, either run Index by UNLs on each node, or consult the &quot;Configure Update Synchronization Service job&quot; for other methods of performing a cluster update.</td>
</tr>
</tbody>
</table>

Important: The UNL file produced by madhubunload for the mpi_memhead table is not sorted by memrecno. Since the Index by UNLs job requires all UNLs to be sorted by memrecno, you must manually sort the mpi_memhead.unl file after using madhubunload.

Configure Update Synchronization Service job
This job is used to enable/disable a configured full text index's dynamic update synchronization service for a node in a clustered environment.

Parameters

Enable this node
Enables the configured full text index's dynamic update synchronization service for the current node. The current node is the one on which this job is run, specified on the first page of the jobs wizard.

Disable this node
Disables the configured full text index's dynamic update synchronization service for the current node. The current node is the one on which this job is run, specified on the first page of the jobs wizard.

Hub Administration jobs

These jobs require no parameters for input or output.

Deploy Hub Configuration
This utility deploys the configuration project to the Hub. This job can be used (instead of the Initiate menu option) to perform the deployment in conjunction with another job.

When this job is executed, the Hub is automatically suspended and resumed, and the Algorithm, Inspector configuration, String data and Flexible Search configuration files are always deployed. When run from the Initiate menu, the option to Check group synchronization is considered selected.
Note: When executing madconfig run_jobset on a job set that includes this job, the entire IBM Initiate Workbench project must be accessible from the Hub.

**Table 21. Deploy Hub Configuration options**

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Command line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configuration Elements</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deploy weight tables</td>
<td></td>
<td>Deploys the weight tables in the selected IBM Initiate Workbench project directory to the Hub. When unchecked (default), no weight tables are deployed.</td>
</tr>
<tr>
<td>Weights folder</td>
<td></td>
<td>Specifies where the weights are located under the selected IBM Initiate Workbench project.</td>
</tr>
<tr>
<td><strong>Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generate DDL instead of creating/dropping tables</td>
<td></td>
<td>Performs database table operations as required to support the configuration.</td>
</tr>
<tr>
<td>Check group synchronization</td>
<td></td>
<td>Checks that the groups listed in the Groups pull-down list on the Groups tab are up to date with the groups defined in the Hub. If this option is checked and the groups do not match, the deployment is aborted.</td>
</tr>
</tbody>
</table>

**Execute Remote Graph**

The Execute Remote Graph job enables you to execute a CloverETL graph on a remote server without having to log into the remote server. This job is intended to re-create the behavior of the madconfig launch_etl utility. For more information about the madconfig utilities, refer to the *IBM Initiate Master Data Service Engine Installation Guide*.

There are two options:

- You can execute a graph that is deployed on a remote server.
- You can copy a graph from IBM Initiate Workbench to a remote server and execute it.

Note: In the job, you must include graph configuration settings, such as connections, parameters, and metadata, that might ordinarily be externalized within a IBM Initiate Workbench project. You can accomplish this by running the CloverETL Export utility against the graph before copying it to the server. Alternatively, you can create a configuration file on the server that includes these settings, or create a configuration file in the local IBM Initiate Workbench project and copy the file to the server with the graph.

To execute multiple graphs in sequence, include multiple Execute Remote Graph jobs in a single job set.

For more information about executing CloverETL graphs, refer to the *IBM Initiate Workbench Clover User’s Guide*.

Note: To enable remote graph execution, you will need to create any directories on the server that are needed for the job to execute successfully, for example, a directory for an output file if the graph is producing one.
Execute a graph that already exists on the remote server

If your graph and optional configuration file are already on the remote server, you can start to build the job. Refer to “The Execute Remote Graph job.”

Copying a local graph to a server for execution

For graphs that you create within IBM Initiate Workbench, you can use external data (outside of the graph file), such as variables and metadata, so that you can share that data with other users and graphs. In such cases, run the CloverETL Export utility against the graph before copying it to the server for execution. The Export utility can embed external data in the graph.

Note: When running a job set via madconfig that includes the Execute Remote Graph job, the graph must already exist on the server.

Graph execution using a configuration file

If you are using external data that cannot be embedded by using the Export utility, you can create a configuration file that includes these definitions and include that file in the job set. The configuration file is a text file that can include definitions of parameters or other information needed to execute the graph.

Defining parameters in a configuration file can make your work more efficient and enable you to share those definitions easily with other users. For example, you can define all the values, paths, and filenames that the graph uses with the help of parameters. Because they are referenced from one file, maintenance is easy if changes occur.

For example, text in the following file defines variables that are used in queries within a graph.

Fri Dec 05 09:34:55 CST 2008
identity=id
hhentity=hh
memattrattr=SEX
memdateattr=BIRTHDT
memnameattr=NAME
memphoneattr=PHONE1
memaddattr=ADDRESS
memidentattr=SSN
relationshipname=live in

The CloverETL documentation includes information about creating external metadata, parameters, and other data, as well as information about externalizing this data from the graph components into a separate file.

The Execute Remote Graph job

Select the options as desired.

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs and Outputs</td>
<td>Specifies that you are working with a graph that exists on the server on which you will be running it.</td>
</tr>
<tr>
<td>Select a graph that already exists on the remote server and execute it.</td>
<td></td>
</tr>
</tbody>
</table>

Table 22. Execute Remote Graph options
### Workbench Description

<table>
<thead>
<tr>
<th>Workbench</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graph File Path</strong></td>
<td>Type the absolute path to the graph file on the server.</td>
</tr>
<tr>
<td><strong>Configuration File Path (Optional)</strong></td>
<td>Type the absolute path to an optional configuration file on the server.</td>
</tr>
<tr>
<td><strong>Select an exported graph from a local directory, copy it to the remote server and execute it.</strong></td>
<td>Specifies that you are working with a graph on the local system. You should have already run the Export utility against the graph.</td>
</tr>
<tr>
<td><strong>Graph File Path</strong></td>
<td>Type or browse to the directory in which the exported graph is stored.</td>
</tr>
<tr>
<td><strong>Configuration File Path (Optional)</strong></td>
<td>Type or browse to the directory in which the optional local configuration file is stored.</td>
</tr>
<tr>
<td><strong>Working directory</strong></td>
<td>Define an optional subdirectory within the server project folder for files associated with the Execute Remote Graph job. A typical directory structure, if the server configuration is similar to the project configuration, is:</td>
</tr>
<tr>
<td></td>
<td><code>hub_instance_path\inst\mpinet_hub_instance_name\work\projectname\work\workingdir</code></td>
</tr>
</tbody>
</table>

### Performance Tuning

<table>
<thead>
<tr>
<th>Performance Tuning</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Java Heap Size</strong></td>
<td>Specify the amount of storage to reserve for processing. The default value is 256 MB.</td>
</tr>
</tbody>
</table>

### Options

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graph parameters</strong></td>
<td>You can add to or override parameters that are defined in the graph or in a configuration file by defining parameter-value pairs in this field. Follow the format that is shown above the field.</td>
</tr>
</tbody>
</table>

When this job is complete, right-click the successful Execute Remote Graph job listed in the Jobs view and select **Get job results** to view information about the graph execution.

#### Exporting a local graph for execution on a remote server:

The Execute Remote Graph job enables you to execute a CloverETL graph on a remote server without having to log into the remote server. If the graph does not already exist on the server, you can export it first.

#### About this task

More detailed information about the Export utility is available in the CloverETL documentation.

#### Procedure

1. From the project directory, right-click the graph node and select **Export** from the context menu.
2. In the Export dialog box, select **Export graphs**. Click **Next**.
3. In the Export graph dialog box:
   a. Navigate to and select the graph to be exported.
   b. Select the options that correspond to the external data that you want to embed in the graph.
   c. (Optional) Specify an Output directory if you want to keep the original, unflattened graph.
   d. Click Finish.

What to do next

You can now build the Execute Remote Graph job.

Get File

Retrieves a file from the Hub. Once the job is finished, right-click the completed job in the Jobs view and select Get Job Results. The text file opens. You can then select all text ([Ctrl][A]), copy it ([Ctrl][C]) and paste it into a text editor to be saved or manipulated as needed.

**Note:** The Get File job cannot be executed in a job set using the madconfig run_jobset command.

<table>
<thead>
<tr>
<th>Table 23. Get File options</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Workbench</strong></td>
</tr>
<tr>
<td>File name</td>
</tr>
<tr>
<td>Encoding</td>
</tr>
</tbody>
</table>

Resume Hub

Resumes end-user Hub interaction.

Suspend Hub

Suspends end-user Hub interaction. This is helpful when you want to prevent user interactions and allow internal Hub processes to continue.

Job Set Templates

Any templates previously created are listed here, enabling you to run (or edit) a job set using the jobs saved in a template. When a template is selected to be included as part of another job set, its individual jobs are added to the job set's list and can be reordered or removed as needed.

Job sets can also be executed from the command line using the madconfig run_jobset command. See "Job set execution" on page 24.
Chapter 4. Configuration editor

This section explains the use of the Member Types, Composite Views, Attribute Types, Relationship Types, Strings, Enumerated Data Types, Informational Sources, Tasks, Linkages, Events, Applications, Groups and Auditing tabs.

Note: Reserved SQL words, such as date and exists, are not permitted as object names in the data dictionary.

You can view differences between Hub configurations in an IBM Initiate Workbench project and those on a Hub by using the Configuration Comparison feature. Refer to "Configuration comparison view" on page 7 for more information.

Note: The Label property is often used in consuming client applications, such as IBM Initiate Inspector. As a general rule, use a meaningful value in all Label properties.

Most of the configuration objects described in this chapter can be added, edited or deleted, but it should be noted that deleting them does not delete corresponding data in the Hub database. If an attribute, for example, is deleted from the dictionary, any data in the database with that particular attribute code is not automatically deleted. For this reason, you should take care not to delete configuration objects that have live data. Doing so could cause fatal errors in the Hub.

Member types

The Member Types view enables you to add, edit and remove member types. Member types identify the “object category” in which data falls (e.g., Person, Provider, Guest, or Organization). There are five objects configurable for a particular Member Type, each having its own tab: Attributes, Entity Types, Composite Views, Sources and Algorithms.

When the Hub Configuration first opens, the Member types view appears by default. The properties in the properties view are listed in Table 24.

Note: The Sources for Selected Attribute box is not populated until sources are defined for the current member type. See "Sources" on page 101 for instructions.

Member types properties

This section contains a list of the property names and descriptions that you can set for a member type. Refer to this list when adding or editing member types.

Table 24. Member types properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Category is used primarily to group for reporting purposes and to act as a placeholder in the system</td>
</tr>
<tr>
<td>Derivation code</td>
<td>The code for the associated derived data specification or strategy. This code maps the member type to a defined strategy for creating derived data. The code must be defined in MPI_dvdhead.</td>
</tr>
</tbody>
</table>
Adding a member type

To add a member type, you must define both the member type definition and the associated derived data strategy.

Procedure

1. In the Member Types view, click Add beside the Member type pull-down list. A new member type is created with default property values.
2. In the Properties view, set the appropriate values for each property. Refer to the table above for property descriptions.

   \textbf{Note:} The information entered in the Derived Data section (Name, Code, and Library Code) must be defined in MPI_dvdhead. Refer to that table for exact values.
3. Save the project.

Editing a member type

To edit the properties of an existing member type, follow the steps below.

Procedure

1. In the Member Types view, select the member type you want to edit from the Member type pull-down list.
2. In the Properties view, make any changes needed.
3. Save the project.

Deleting a member type

Use caution when deleting member types, as once the project is saved, they cannot be restored by the use of Ctrl+Z or Edit > Undo after deletion. Deleting member types does not delete member data associated with those member types.
Procedure
1. In the Member Types view, select the member type you want to delete from the Member Type pull-down list.
2. Click **Remove** beside the Member Type pull-down list.
3. Save the project.

Attributes
The Attributes view enables you to easily add attributes to the Initiate software, in order to associate the attributes with segments. You reach the Attributes view by first clicking the Member Types tab, and then clicking the Attributes tab.

Attributes are stored in various database tables.

Attribute properties
This section contains a list of the property names and descriptions that you can set for a member type’s attributes. Refer to this list when adding or editing attributes.

*Table 25. Attributes properties*

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attrrecno</td>
<td>A read-only field containing the ATTRRECNO field from the Hub database.</td>
</tr>
<tr>
<td>Code</td>
<td>The code used to define the attribute. Avoid using underscore characters for attributes whose Storage type is “external.”</td>
</tr>
<tr>
<td>Description</td>
<td>The description of the attribute.</td>
</tr>
<tr>
<td>Enumerated data type code</td>
<td>Enumerated Data Type associated with this attribute. For example, Marital Status may have an EDT of “married.”</td>
</tr>
<tr>
<td>Label</td>
<td>A label for the attribute.</td>
</tr>
<tr>
<td>Maximum active values</td>
<td>The number in this field controls status assignment for updated values. For example, if this number is set to 1, only one attribute value of this type will have an active status. Any updates to that value result in the new value being active and the previous value becoming inactive. A value of 0 indicates an unlimited number of active values will be maintained.</td>
</tr>
<tr>
<td>Maximum existing values</td>
<td>This number indicates how many updated values are stored in history. Whatever number this is set to, the software stores that number of values—regardless of status—in history. For example, if it is set to 2 and there are two attribute values currently stored, a third value causes the earliest value to be removed (trimmed) from the database. A value of 0 indicates an unlimited number of active values will be maintained.</td>
</tr>
<tr>
<td>Member status filter</td>
<td>Member status filter controls which members are cross-matched to create entities for this type; Active, Merged, Overlay, Deleted or Fictitious. For example, if Active and Merged are checked, then member records with a status Active or Merged can have their attribute values modified.</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the attribute.</td>
</tr>
</tbody>
</table>
Table 25. Attributes properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage type</td>
<td>The following are valid values for the storage type:</td>
</tr>
<tr>
<td></td>
<td><strong>Normal</strong> if the attribute is stored in the Hub with other data from this</td>
</tr>
<tr>
<td></td>
<td>source.</td>
</tr>
<tr>
<td></td>
<td><strong>Virtual</strong> if the attribute is not stored in the Hub database.</td>
</tr>
<tr>
<td></td>
<td><strong>Hub Controlled</strong> if the attribute is to be controlled by Initiate software,</td>
</tr>
<tr>
<td></td>
<td>Hub controlled attributes are often defined to support users of Initiate</td>
</tr>
<tr>
<td></td>
<td>software during data remediation activities or for customers who wish to</td>
</tr>
<tr>
<td></td>
<td>track and use attributes that are not stored in a source system. In normal</td>
</tr>
<tr>
<td></td>
<td>implementations, once a member is merged or logically deleted, their</td>
</tr>
<tr>
<td></td>
<td>attributes cannot be modified. Additionally, attributes that are in a</td>
</tr>
<tr>
<td></td>
<td>shadow state cannot be modified. However, if an attribute is defined as</td>
</tr>
<tr>
<td></td>
<td>“controlled,” and thus is independent of a source system, the usual</td>
</tr>
<tr>
<td></td>
<td>attribute validation does not apply and modification is allowed.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: If you are adding Hub Controlled attributes, an HCA composite</td>
</tr>
<tr>
<td></td>
<td>view is automatically created. Once the HCA is created, any Hub Controlled</td>
</tr>
<tr>
<td></td>
<td>attributes added via the Add Attribute function will automatically be</td>
</tr>
<tr>
<td></td>
<td>added to the view. IBM Initiate Workbench will maintain the attributes for</td>
</tr>
<tr>
<td></td>
<td>you in the HCA view (e.g., if you delete the attribute from the Attributes</td>
</tr>
<tr>
<td></td>
<td>view, the attribute is removed from the HCA view). Also, if you want to</td>
</tr>
<tr>
<td></td>
<td>be able to update Hub-controlled attributes in IBM Initiate Inspector, you</td>
</tr>
<tr>
<td></td>
<td>must select the Hub Controlled checkbox and select Member Status Filter</td>
</tr>
<tr>
<td></td>
<td>settings. <strong>Note</strong>: It is not necessary to define the attribute for all</td>
</tr>
<tr>
<td>Type</td>
<td>The member type associated with the attribute. Each attribute type consists</td>
</tr>
<tr>
<td></td>
<td>of one or more data fields that describe the individual components of the</td>
</tr>
<tr>
<td></td>
<td>attribute. Refer to “Member types” on page 89 for information on managing</td>
</tr>
<tr>
<td></td>
<td>attribute types.</td>
</tr>
</tbody>
</table>

Adding an attribute
Add an attribute for a member type when you need to track a particular type of data, such as shoe size for a person member, or ISBN for a publication member.

Procedure
1. In the Attributes view, click Add. A new attribute with default properties is created.
2. In the Properties view, set the appropriate values for each property. Refer to the table above for property descriptions.
3. In the Sources for selected Attribute list, check each source name from which data for this attribute will be available.
4. Save the project.

Editing an attribute
Most of the properties for an attribute can be edited by following this process.
Procedure
1. In the Attributes view, highlight the desired attribute.
2. In the Properties view, make any changes necessary.
3. Save the project.

Deleting an attribute
Use caution when deleting attributes, as once the project is saved, they cannot be restored by the use of Ctrl+Z or Edit > Undo after deletion. Deleting attributes does not delete member attribute data associated with those attributes.

About this task
Note: Deleting an attribute may affect your comparison scores.

Procedure
1. In the Member Types: Attributes view, highlight the desired attribute.
2. Click Remove on the Attributes tab.
3. Save the project.

Entity types
The Entity Types view enables management of your entity types (e.g., Identity or Household). An entity type allows for distinction between the way members are viewed and linked within Master Data Engine software; thus each type is associated with a specific algorithm configuration. Certain information managed through this view controls Entity Manager behavior (Entity Manager behavior is described in the IBM Initiate Master Data Service Hub Overview document).

Functionality is built into this component to help manage entity relationships by controlling whether same source auto-linking is allowed. For example, we have Member A and Member B—both from Source 1—who do not score above the auto-link threshold for Source 1-to-Source 1 comparisons. However, in a Source 1-to-Source 2 comparison, Member C in Source 2 gets a favorable comparison above the auto-link threshold with both Members A and B. This results in erroneously gluing A and B together in an entity with C. By turning off same-source auto-linking, you can help avoid the creation of erroneous entities.

You reach the Entity Types view by first clicking the Member Types tab, and then clicking the Entity Types tab.

Entity type properties
This section contains a list of the property names and descriptions that you can set for a member type's entity type. Refer to this list when adding or editing an entity type.

Table 26. Entity type properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow same source linking</td>
<td>Enables records from the same source that score above the auto-link threshold to be automatically linked with a common Enterprise ID. (Records from the same source can still be manually linked.) The default setting is false.</td>
</tr>
<tr>
<td>Asynchronous</td>
<td>Indicates whether asynchronous processing is enabled for the Entity Manager.</td>
</tr>
</tbody>
</table>
**Table 26. Entity type properties (continued)**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Category is used primarily to group for reporting purposes and to act as a placeholder in the system.</td>
</tr>
<tr>
<td>Comparison code</td>
<td>Defines the comparison algorithm used for this entity. Valid codes can be found in MPI_cmphead.</td>
</tr>
<tr>
<td>Comparison description</td>
<td>Brief description of the comparison algorithm used for this entity.</td>
</tr>
<tr>
<td>Comparison name</td>
<td>Name for the comparison algorithm used for this entity.</td>
</tr>
<tr>
<td>Description</td>
<td>Brief description of the entity type.</td>
</tr>
<tr>
<td>Entity manager polling interval</td>
<td>This entry controls when the Entity Manager “wakes up” to check for flagged MEMRECNOs. Note that the Entity Manager only “sleeps” when it is empty. If flagged MEMRECNOs continue to be present, the Entity Manager continues to process. If an error occurs during entity management this value is also used as the sleep duration before processing is resumed by the entity manager instance. This value is in seconds, and the minimum value is 1.</td>
</tr>
<tr>
<td>Entity manager work unit</td>
<td>This entry defines how many records come off the Entity Manager work queue in one unit.</td>
</tr>
<tr>
<td>Entity type</td>
<td>The type of entity (e.g., ID for Identity or HH for Household).</td>
</tr>
<tr>
<td>Enttypeno</td>
<td>A read-only field containing the ENTTYPENO field from the Hub database.</td>
</tr>
<tr>
<td>Label</td>
<td>An expanded string or description of the status. This entry is displayed on end-user client screens.</td>
</tr>
<tr>
<td>Maximum bucket role</td>
<td>This is used to set a bucket role value range to be used when doing a member match (cross matching by an entity manager, BXM, or MemMatch interaction). It is a subset of the full range of values that are used for a member search, and is a performance tuning setting to allow for faster matches. Bucket roles outside this range are used for searches but not for matches.</td>
</tr>
<tr>
<td>Maximum candidate count for dynamic frequencies</td>
<td>This property is used to enable dynamic frequency-based bucketing. Dynamic frequency-based bucketing is a run time process. When a frequent bucket is identified, an entry is added in the database. This newly added bucket value is considered during subsequent member data derivation. Dynamic frequency-based bucketing is triggered when the number of candidates identified during candidate selection exceeds the setting of the &quot;Maximum candidate count for dynamic frequencies&quot; property. Valid input is an integer of 0 - 2147483647. The default of 0 means that dynamic frequency-based bucketing is not enabled.</td>
</tr>
<tr>
<td>Member Status Filter</td>
<td>Member status filter controls which members are cross-matched to create entities for this type; Active, Merged, Overlay, or Deleted. The “F” status is commonly used when an implementation has records that are invalid, but for various reasons choose to keep in the system. For example, a pharmacy that fills prescriptions for both humans and animals may want to keep records of the animal prescriptions and be able to search for those records, but they do not want to actually link those records to entities made up of human records. To prevent linking for members with a status of “F,” do not specify “F” as a member status filter.</td>
</tr>
</tbody>
</table>
Table 26. Entity type properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum bucket role</td>
<td>This is used to set a bucket role value range to be used when doing a member match (cross matching by an entity manager, BXM, or MemMatch interaction). It is a subset of the full range of values that are used for a member search, and is a performance tuning setting to allow for faster matches. Bucket roles outside this range are used for searches but not for matches.</td>
</tr>
<tr>
<td>Transitive</td>
<td>Indicates whether glue members are used in matching when placing members into entities. The default is True. When set to True, if Member A matches Member B, and Member B matches Member C, it is assumed that Member A matches Member C and they are all put into the same entity. When set to False, this assumption is not made, so every member in an entity is guaranteed to match above the AL threshold with every other member in the entity.</td>
</tr>
<tr>
<td>Uses an input queue</td>
<td>Indicates whether this entity type has an input queue. Input queues come into play when the entity is set to link and create tasks. If you are using synchronous entity management or not using persistent entity management, set this to false.</td>
</tr>
<tr>
<td>Uses linkage history</td>
<td>Indicates whether this entity type stores Enterprise ID assignment history.</td>
</tr>
<tr>
<td>Uses linkages</td>
<td>Indicates whether this entity type allows linkages of records.</td>
</tr>
<tr>
<td>Uses notes</td>
<td>Indicates whether this entity type stores notes about the records.</td>
</tr>
<tr>
<td>Uses rules</td>
<td>Indicates whether this entity type sets and stores identity rules between records in an entity.</td>
</tr>
<tr>
<td>Uses tasks</td>
<td>Indicates whether this entity type uses tasks.</td>
</tr>
</tbody>
</table>

Adding an entity type
Add an entity type when you need to create a new distinction between the way members are viewed and linked.

About this task
To add an entity type, follow the steps below.

Procedure
1. In the Entity Types view, click Add. A new entity type is created with default properties.
2. In the Properties view, provide the appropriate values for each property.

   Note: After adding a new entity, you will need to set the Clerical Review Threshold and Autolink Thresholds for each pair of sources. This is done in the Algorithm Editor’s Thresholds and Weight Properties tab. Refer to Chapter 5 “Algorithm editor,” on page 153 for instructions.
3. Save the project.

Editing an entity type
Most of the properties for an entity type can be edited by following this process.
**Procedure**
1. In the Entities view, select the desired entity from the table.
2. In the Properties view, make any necessary changes.
3. Save the project.

**Removing an entity type**
Use caution when deleting entity types, as once the project is saved, they cannot be restored by the use of Ctrl+Z or **Edit > Undo** after deletion. Deleting entity types does not delete data associated with those entity types.

**Procedure**
1. In the Member Types:Entity Types view, select the unwanted entity type from the table.
2. Click **Remove** beside the Entity Types grid.
3. Save the project.

**Source priority properties**
Use these property names and descriptions when setting an entity type's source priority during an add or edit operation.

*Table 27. Source priorities for entity types*

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>A positive integer to represent the priority of the source. Higher numbers have higher priority. The default value is 1 (the lowest priority). A value of 0 will result in no members from the entity being linked.</td>
</tr>
<tr>
<td>Source</td>
<td>The name of the source for which the priority is being defined. This field is required.</td>
</tr>
<tr>
<td>Trusted</td>
<td>Indicates whether the priority value should be ignored for trusted sources. The default is false.</td>
</tr>
</tbody>
</table>

**Source priorities for entity types**
In some cases, users may wish to define priorities for sources within an entity. Currently this is implemented to support BestMatch linking capability and custom coded handlers; no additional out-of-the-box functionality is available to make use of source priorities. Entity types do not require a priority.

**Adding a source priority for an entity type:**
To add a source priority for an entity type, follow the steps below.

**Procedure**
1. Select the entity type for which you wish to create a source priority.
2. In the “Source priorities for selected entity type” group box, click **Add**. A new entry is created with default properties.
3. In the Properties view, provide the appropriate values for each property.

**Editing a source priority for an entity type:**
To edit a source priority for an entity type, follow the steps below.
Procedure
1. In the Entities view, select the entity from the table, then select the source priority to be changed.
2. In the “Source priorities for selected entity type” group box, make the desired changes.
3. Save the project.

Removing a source priority for an entity type:

To remove a source priority for an entity type, follow the steps below.

Procedure
1. In the Entities view, select the entity from the table, then select the unwanted source priority.
2. Click Remove in the “Source priorities for selected entity type” group box.
3. Save the project.

Composite Views

Use the functionality in this component to define how current attributes display in the Enterprise Viewer and Composer applications.

The composite view represents your defined “complete picture” of a member. Configuration of composite views establishes the rules that control the behavior and display of member attribute data in Initiate applications.

You can implement certain security measures by defining the attributes that are returned to the display. For example, your member data may be made up of Name, Address, Phone, and Social Security number. To avoid exposing the Social Security number to your end users, you can filter that attribute out.

The following describes some options for composite views:

**EMCA**
Entity Most Current Attribute. This display takes the attributes from any record within a defined linkage (records that share a common Enterprise ID) and combines the attributes so that the view of the member is a conglomeration of attributes from your various source systems.

**MMCA**
Member Most Current Attribute. This display shows the most current attributes for a desired member and does not take into consideration any enterprise level information.

**Trusted**
This view displays the most current attributes available from the specified source. This view can be used if an organization has great confidence that the data from a particular source is accurate or they wish to restrict access to all but one source.

**Controlled**
This view displays a selection of various attributes from multiple specified sources. For example, you can specify that name and address are retrieved from Source A, phone and birth date from Source B, and Social Security number and address from Source C.

**Restricted**
A restricted composite view shows only EMCA views. Users assigned this
composite view in their user profile will not have the option of seeing source-specific columns. A restricted view is often implemented in environments where security policies require controlled access to certain data. One example is in a hospital where an employee in the billing department needs to see a member name and insurance information, but they are not legally allowed to view specific medical data.

**Handler composite view**

Handler composite view attributes are post-processed by the callout mechanism using a built-in callout handler shipped with the Hub. Compared with Standard Composite Views, they give the user a finer-grained level of control in their configuration. Note that Handler Composite Views will only fire if the Hub is setup to use the mpicbjava.dll or mpicbdotnet.dll callback library.

**Record Status**

The following are possible record status values:

- **Active** - represents the most current value of this attribute.
- **Inactive** - represents an attribute that has been updated.
- **Shadow** - after an attribute is updated in Initiate software, but before it is confirmed by the source system, it is assigned a value of shadow. A shadow attribute becomes active when an update message is received by Master Data Engine software that the change has been made in the source system.
- **Deleted** – represents an attribute that has been logically deleted. A logically deleted attribute is maintained in the database for historical purposes, but is not used in comparisons against other member records.

When defining composite views, the Initiate software can be configured to use either the Initiate event date or a Source event date to determine the most current attribute to return for a member. This setting becomes important in environments where a particular source feeds data to the Initiate software in a batch or delayed mode and guards against out-of-date information being displayed as the most current attribute value in Enterprise Viewer.

If you do not create composite views, Enterprise Viewer defaults to an enterprise view.

To open the Composite Views view, click the Member Types tab, select the desired Member Type from the pull-down list, and then click the Composite Views sub-tab.

**Composite view properties**

This section contains a list of the property names and descriptions that you can set for a member type’s composite views. Refer to this list when adding or editing a composite view. The Name is used for search criteria when searching for a view.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cvwrecno</td>
<td>A read-only field containing the CVWRECNO field from the Hub database.</td>
</tr>
<tr>
<td>Description</td>
<td>A brief description of the view.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Handler function</strong></td>
<td>For Handler composite views, identifies the function to be executed when the composite view fires. Valid choices are Most Current Attribute, Source Priority, and Unique Attributes.</td>
</tr>
<tr>
<td><strong>Most Current Attribute</strong></td>
<td>In the MemRowList returned to the caller, each attrCode's MemAttrRows are returned in descending maudrecno order.</td>
</tr>
<tr>
<td><strong>Source Priority</strong></td>
<td>The source priority values (defined within mpi_entxsrc) are considered when returning attribute values. If an attribute belongs to a high-priority source, it is returned before attributes from lower-priority sources are. If not listed within mpi_entxsrc, a source is considered to have priority=0. Note also that you cannot specify a Source Priority value of 0 explicitly within a row in mpi_entxsrc, otherwise problems will occur when running bulk linkages.</td>
</tr>
<tr>
<td><strong>Unique Attributes</strong></td>
<td>Only unique attribute values are returned to the caller when this composite view function is specified. Uniqueness is determined by comparing all of the attribute-specific fields to each other; any duplicates are filtered out. For instance, if a Name attribute has two fields (Last,First), and an entity has the following accumulation of Name attributes: “Smith, Bob”, “Kennedy, Bob”, “Smith, Robert”, and “Kennedy, Bob”, then only the attribute “Smith, Bob”, “Kennedy, Bob”, and “Smith, Robert” will be returned. Which of the duplicated attributes is preserved (versus which are filtered out) is indeterminate.</td>
</tr>
<tr>
<td><strong>Handler arguments</strong></td>
<td>For Handler composite views, identifies the arguments sent to the function identified in the Handler function property. The arguments vary depending on the Handler function selected. <strong>Not currently used by any handler Composite view function.</strong></td>
</tr>
<tr>
<td><strong>Kind</strong></td>
<td>Identifies whether the view returns Member-centric information or Entity-centric information. If Member is selected, you must define attribute filters. For EMCA or Restricted views, select Entity. MMCA should be set to Member. Trusted and Controlled can be set to either Member or Entity.</td>
</tr>
<tr>
<td><strong>Most current attribute date type</strong></td>
<td>This determines which system event date to use in determining the most current attribute value for a member; Initiate or Source. This setting becomes important in environments where a particular source feeds data to the Initiate software in a batch or delayed mode. If you have a source that sends data to the Initiate software in a batch or delayed mode, you may want to set the date type to Source to prevent older data from being used as the most current value. If your sources feed the Initiate software in real-time, then using the Initiate date type is appropriate. This property is only available in Standard and Handler composite views.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>The name of the view; appears on all composite view selection menus.</td>
</tr>
</tbody>
</table>
Table 28. Composite view properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of current attributes to return</td>
<td>Determines the depth of the most current attributes to return; 0 (zero) means both current and non-MCA attributes are returned, 1 means only the most recent attributes known are returned, 2 returns the two most current attributes for the known attributes, and so on.</td>
</tr>
<tr>
<td>Number of results to return</td>
<td>Defines the maximum number of member matches to return. The default of 0 (zero) returns an unlimited number of rows.</td>
</tr>
<tr>
<td>Record status filter</td>
<td>This determines whether the records returned are restricted by their status (Active, Inactive, Shadow, or Deleted). Only those records with the designated status are returned.</td>
</tr>
<tr>
<td>Restricted</td>
<td>Determines whether records returned are restricted. (See above for definition.)</td>
</tr>
</tbody>
</table>
| Type                                            | Indicates whether the view is a standard or handler composite view.  
  • Standard composite views are evaluated by the Hub. Most composite views are Standard.  
  • Handler composite view attributes are post-processed by the callout mechanism using a built-in callout handler shipped with the Hub. Compared with Standard Composite Views, they give the user a finer-grained level of control in their configuration. Note that Handler Composite Views will only 'fire' if the Hub is setup to use the mpicbjava.dll or mpicbdotnet.dll callback library. |

Adding a composite view

Add a new composite view when you need to define a way of displaying attribute information.

Procedure

1. In the Composite Views view, click Add. A new Composite View is created with default parameters.
2. Provide appropriate values for the properties listed in the Properties view. Refer to the table above for descriptions of each property.
3. In the Filters for selected Composite View section, click Add. The New Composite View Filter dialog opens.
4. Select the Attribute to return in the segment from the pull-down list.
5. Select the Source from which the attribute value will be returned. If you leave the Source field blank, IBM Initiate Workbench adds one row for each definitional source for the member type in the MPI_CVWXSEG table.
6. Specifying a Priority enables more control in displaying MCA values for an attribute. For example, if you set the following for phone number:  
   Source A – Guest Home Telephone, Priority 1  
   Source C – Guest Home Telephone, Priority 2  
   Source B – Guest Home Telephone, Priority 3
   In this case, when retrieving a member, the home phone MCA value will come from Source A. However, if Source A did not have a phone number stored for this member, the value would come from Source C, and so on. More than one source can share the same priority level for an attribute, in which case the mducrecno/timestamps determine the priority.
7. Specify the **Handler function code** if different from the parent Handler Composite View's Handler function. If the parent composite view does not have a handler function specified, any attributes that don’t have a specific filter defined will not be returned when using that view. This would be valuable if you wanted a composite view that only returned a very sparse or focused amount of data.

8. The **Handler function arguments** string is not currently used by any handler Composite view function.

9. Click **OK**. The attribute appears in the *Filters for selected Composite View* section as shown in the example above.

10. Repeat steps 3-9 as necessary for each attribute you want to add.

**Editing a composite view**

Most of the properties for a composite view can be edited by following this process.

**Procedure**

1. In the Composite Views view, highlight the view you want to edit.
2. In the Properties view, set the appropriate values for each property. Save the project.

**Deleting a composite view**

Use caution when deleting composite views, as once the project is saved, they cannot be restored by the use of Ctrl+Z or **Edit > Undo** after deletion.

**Procedure**

1. In the Composite Views view, highlight the view you want to edit.
2. Click **Remove**. All attributes and filters are automatically removed as well.
3. Save the project.

**Setting group access for composite views**

Group access determines which users can access a composite view and the data it contains.

**Procedure**

1. Click the Groups tab, then click the Composite Views tab. The currently defined composite views are shown.
2. Select the desired LDAP group from the Group pull-down list.
3. Indicate the Composite Views to which you wish this group to have access. More information can be found in [Groups](#) on page 145.

**Sources**

The Sources view enables you to add and manage information about the sources that interact with Initiate software. There are typically two types of sources with which the Initiate software interacts: *definitional* and *informational*.

- A definitional source is one in which members (records) are created and usually updated (e.g., a registration system). Initiate software, via the Outbound Message-Based Transaction Service, can send updates to a definitional source.

- Informational sources are thought of as “look up” data sources, typically providing “legal” values used to identify certain member attributes (e.g., credit card number/issuer or frequent customer number/issuer). The Initiate software does not send updates to an informational source.
**Sources properties**

The Sources tab within the Member Types view enables you to add and remove definitional sources. You reach the Sources view by first clicking the Member Types tab, and then clicking the Sources tab.

**Table 29. Sources properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Entity Priority</td>
<td>Enables you to prioritize entity management by source. The allowable values are 0 - 32767. The lower the number, the higher the entity manager processing priority. For example, a source with an Entity Priority of 1 will be processed before a source with priority 2. If you do not wish to enable entity management priority, set all of your sources to the same value, such as 0. When entity management by source is disabled, the default behavior is for entity management processing to be done in create time order (caudRecno). Note that setting entity management source priority should not be confused with source priority, which is used to determine trusted source linking. With regards to trusted source linking, the entity manager determines which members must be re-crossmatched if a trusted source record links or unlinks. These members have entity records created and inserted into the entity manager queue. When the records are created, they will use the same entity management priority as their linked trusted source member. For example, if a record from Source A is linked to trusted source record B, B's priority is used for the entity manager re-queue. Default: 100</td>
</tr>
<tr>
<td>Enable Review Identifier Task</td>
<td>Indicates whether records from this source will be compared for Review Identifier tasks.</td>
</tr>
<tr>
<td>Is virtual?</td>
<td>Indicates if the source is virtual. Member attributes from virtual sources are not written to the Hub database, but do participate in derived data. By setting this field to 'Y,' you are indicating that all attributes coming from this source should be treated as virtual and are not available for display in task and member searches. A 'Y' setting in MPI_srchead overrides the isVirtual setting of 'N' in MPI_SegAttr (virtual attributes). See Note about virtual settings below for additional information.</td>
</tr>
<tr>
<td>Physical code</td>
<td>A physical code provides a way to group sources, but does not affect Hub processing. For example, a hospital may have separate source systems for the laboratory, registration desk, and pharmacy. This code enables a grouping of these systems under one physical umbrella, yet retains a different source code identifier for each system.</td>
</tr>
<tr>
<td>Potential overlay comparison code</td>
<td>The comparison function used for overlay comparison. This code is found in MPI_cmphead. If no function is selected, potential overlay comparison is disabled.</td>
</tr>
<tr>
<td>Potential overlay score</td>
<td>The overlay comparison score; a negative number. Records from this source scoring below this number are placed in Potential Overlay tasks. The software assumes a decimal. For example if you enter -10, the software converts this to -1.0.</td>
</tr>
</tbody>
</table>
Table 29. Sources properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record status</td>
<td>Indicates whether the source is Active (in use) or Inactive. Selecting Inactive will not prevent source-to-source comparisons against records from this source. An inactive status will prevent inserts (adds), updates, and deletions to an inactive source. If a source is marked inactive and an interaction is attempted, an error is returned.</td>
</tr>
<tr>
<td>Source code</td>
<td>This code is used as a prefix on all source-assigned record numbers from this source in the Initiate software (e.g., RMC:880189).</td>
</tr>
<tr>
<td>Source name</td>
<td>Name of this source.</td>
</tr>
<tr>
<td>Srcrecno</td>
<td>A read-only field containing the SRRECNO field from the Hub database.</td>
</tr>
</tbody>
</table>

Note about virtual settings: Issues such as limitations on data storage capacity may lead to the use of virtual sources—member data not being stored in the Hub database or only in derived data form. Keep in mind when marking sources (or attributes) as virtual:

- Because virtual attributes are not stored in the core attribute tables, attribute history is not maintained. In order to search or compare on historical attributes, you need to send in all values (current and historical) for a virtual attribute on every update.
- Because the data is stored in derived format only, you will not be able to easily reload the data for upgrades. During an upgrade, you must reload the virtual data from the source.

Note: The difference between virtual sources and external sources is that data from a virtual source is stored in derived data. Data from an external source is not stored anywhere in the Initiate software.

Adding a definitional source
Add a definitional source when you want access to member records maintained within that source. Adding a source is a two-step process: 1) adding the source information (stored in MPI_srchead) and then 2) adding the comparison scores for this source and other sources interacting with the Initiate software (stored in MPI_srcxsrsr).

Procedure
1. Click on the Member Types tab and select the desired member type from the pull-down list.
2. Click on the Sources tab to open the sources view.
3. Click Add. A new source is created with default property values.
4. In the Properties view, provide appropriate values for each property. Refer to the table above for property descriptions.
5. Save the project.

Results
When sources are added, they are automatically associated with all of the existing member type attributes.
To disassociate individual attributes, remove the check mark from the attribute in the Attributes for selected source box.

To disassociate multiple attributes at a time, go to the Attributes tab, select the desired attributes (use the Shift or Ctrl key while clicking to select multiple attributes), and then remove the checkbox from the source name in the Sources for selected attribute box.

Editing a source
Most of the properties for a source can be edited by following this process.

Procedure
1. On the Sources tab, highlight the Source.
2. Make any change necessary in the Properties view.
3. Save the project.

Deleting a source
Use caution when deleting sources, as once the project is saved, they cannot be restored by the use of Ctrl+Z or Edit > Undo after deletion. Any Clerical Review and AutoLink thresholds configured for the deleted source must be deleted also. Deleting sources does not delete the source data associated with those sources.

About this task
Note: Deleting a source may have adverse affects on some Hub processes. Consider, instead, setting the source’s Record status to Inactive.

Procedure
To delete source comparison scores:

Note: Deleting a source does not automatically delete its comparison scores.
1. Open the algorithm for which comparison scores were created for the unwanted source.
2. Click the Thresholds and Weight Properties tab.
3. Select the entity type for which comparison scores were created for the unwanted source.
4. For each source pair that includes the unwanted source, click Delete.
5. Save the project.

To delete the source:

Note: Deleting a source does not automatically delete its comparison scores.
6. Highlight the unwanted source from the Sources tab, and then click Remove.
7. Save the project.

Algorithms

About this task
The Algorithms tab enables you to create or identify the active algorithm that the Hub uses to process comparisons. The algorithms shown on this tab have no properties in the Properties view; any properties displayed there pertain to the previous configuration artifact viewed. Algorithms are configured using the Algorithm Editor. Refer to Chapter 5, "Algorithm editor," on page 153.
Adding an algorithm
Add an algorithm when you need to create tasks and support search functionality for a member type.

About this task
Creating a new algorithm automatically associates it with the current member type. Refer to Chapter 5, “Algorithm editor,” on page 153 for instructions on using the Algorithm Editor to create your new algorithm.

Procedure
1. In the Algorithms view, click Add. You are prompted to save the Hub configuration first. A new algorithm is created with default properties. New algorithms are always Inactive.
2. The Algorithm Editor opens with the new (empty) algorithm loaded.
3. Save the project. The algorithm is not uploaded to the Hub until it is deployed. Refer to “Deploying a Hub configuration” on page 17 for instructions.

Setting an algorithm to active
Set an algorithm to active when it is ready to be used for task creation and searching. Only one algorithm can be active per member type on a Hub; making one algorithm active sets all other algorithms for the current member type to inactive.

Procedure
1. In the Algorithms view, select the desired algorithm.
2. To set this algorithm as the active algorithm, click Make Active.
3. Save the project.

Editing an algorithm
Algorithms are edited using the Algorithm Editor.

About this task
Refer to Chapter 5, “Algorithm editor,” on page 153 for details on using the Algorithm Editor.

Procedure
1. In the Algorithms view, select the desired algorithm.
2. Click Edit or double-click the desired .alg file in the Navigator pane. The algorithm opens in the Algorithm Editor.
3. Save the project when editing is completed.

Copying an algorithm within a project
You can make a duplicate of an algorithm to make minor adjustments or to have a starting point for a similar algorithm.

Procedure
1. In the Algorithms view, select the desired algorithm.
2. Click Copy. You are warned that this operation cannot be undone. The algorithm is copied and named in the format Copy_x_of_original_algorithm_name.alg. You can change the filename as desired by clicking on it in the Algorithms tab and typing a new name in the
Name column. A corresponding .alg file is created in the Navigator. Clicking Edit will open the copy in the Algorithm Editor.

3. Save the project.

Copying an algorithm between projects
You can copy an algorithm from one project to another for testing purposes.

Procedure
1. In the Algorithms view, select the desired algorithm in the source project.
2. Right-click the algorithm in the Navigator pane and select Copy.
3. Right-click the target project name and select Paste. If the target project has an algorithm with the same name, you are asked to verify that you wish to overwrite the existing algorithm. You can then associate this algorithm to a member type by following the instructions in "Associating an existing algorithm with the current member type" below.
4. Save the project.

Disassociating or deleting an algorithm
If you no longer want an algorithm, you can disassociate it from its member type for possible use later, or delete it altogether.

Procedure
1. In the Algorithms view, select the unwanted algorithm.
2. Click Remove. The Remove Algorithm dialog opens. You have the option to permanently delete the algorithm or disassociate the algorithm from its current member type.
   • To disassociate the algorithm from its member type and retain the algorithm for use later, click Don't delete the Algorithm. (Pressing the Enter key executes this option.)
   • To permanently delete the algorithm from the database, click Delete the Algorithm.
   CAUTION: This operation cannot be reversed through the Edit > Undo or Ctrl+Z operation.
3. If the removed algorithm had been the active algorithm for the current member type, set another algorithm to Active. See “Setting an algorithm to active” on page 105.
4. Save the project.

Associating an existing algorithm with the current member type
Algorithms that have been created previously can be used with another member type. This can be useful when testing an existing algorithm (or a copy of an algorithm) with a new member type.

Procedure
1. Navigate to the Member Type to which you wish to associate the algorithm.
3. From the list, select the algorithm to add to this member type.
4. Click OK.
5. Save the project.
Attribute types

Attribute types (segments) coincide with the Initiate data schema to define Hub behavior and member information. Two types of Attribute Types are available: Member Attribute Types and Relationship Attribute Types.

**Member Attribute Types**

A set of pre-defined (“fixed”) attribute types are packaged with the Master Data Engine software; these attribute types are defined in the *IBM Initiate Master Data Service SDK Reference for Java and Web Services*. To allow for flexibility at customer sites, implementers can add member-attribute implementation-defined attribute types.

**Implementation-defined attribute types**

As mentioned above, there are pre-defined attribute types created and shipped with the Initiate software. For example MemName, MemAddr, MemIdent are all pre-defined attribute types. Implementation-defined attribute types are those created at the time of the hub’s implementation—by Initiate Services or Partner implementers—and therefore are not associated with a generated class.

**Relationship Attribute Types**

Relationship attribute types are attribute types that are specific to relationships. All attribute type names must be unique; an attribute type cannot be both a member attribute type and a relationship attribute type. Once relationship attribute types are defined, they can be associated with relationship type attributes on the Relationship types tab. Refer to “Relationship types” on page 122 for more information.

**Attribute type properties**

This section contains a list of the property names and descriptions that you can set for an attribute type. Refer to this list when adding or editing an attribute type.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>Attribute type (segment) code name.</td>
</tr>
<tr>
<td>Segrecno</td>
<td>A read-only field containing the SEGRECNO field from the Hub database.</td>
</tr>
<tr>
<td>Table name</td>
<td>Name of the database table which stores the attribute type data.</td>
</tr>
<tr>
<td></td>
<td>Workbench does not allow creation of uppercase table names due to an</td>
</tr>
<tr>
<td></td>
<td>incompatibility with engine utility support.</td>
</tr>
<tr>
<td>Uses history</td>
<td>Indicates whether history will be kept for the selected attribute type.</td>
</tr>
<tr>
<td></td>
<td>This property can also be set on the History tab. (See “History” on page 110.)</td>
</tr>
</tbody>
</table>
Attribute field properties

This section contains a list of the property names and descriptions that you can set for an attribute field. Refer to this list when adding or editing an attribute field.

Table 31. Attribute field properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field data type</td>
<td>Type of data stored in this attribute field.</td>
</tr>
<tr>
<td>Field label</td>
<td>A label for the field.</td>
</tr>
<tr>
<td>Field length</td>
<td>Maximum length of the data in the field. This property does not apply to numeric or date type fields.</td>
</tr>
<tr>
<td>Field sequence number</td>
<td>The sequence of this field within the attribute.</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the attribute field. This value must be all lower case.</td>
</tr>
<tr>
<td>Required</td>
<td>Indicates whether the field is required. This field is available for member attribute types only.</td>
</tr>
<tr>
<td>Virtual</td>
<td>Indicates whether this is a virtual attribute. Attributes defined as virtual are not stored in the Initiate database, but do participate in data derivation. By setting this to “Yes,” this attribute will not be available for display and use in task and member search functionality. As well, a history of the attribute is not maintained in the Initiate database.</td>
</tr>
</tbody>
</table>

Displaying attribute types

Attribute types must be unique; an attribute type code cannot be used for both a Member Attribute Type and Relationship Attribute Type.

About this task

To view an attribute type's properties and fields, follow this process.

Procedure

1. Select the desired Type from the Types pull-down list.
2. Pick the Attribute type code from the Attribute type pull-down list.

Results


Adding an attribute type

The Initiate Master Data Engine enables implementers to define custom attribute types for customers. Use the Attribute Types tab to view and edit attribute types.

About this task

While the specific steps for adding an attribute type are provided later in this section, there are a couple items to note about the process.

- You must choose whether to define Member attribute types or Relationship attribute types. Member attribute types define the attributes associated with regular members in the system. Relationship attribute types are specific to relationship types used by the relationship linker.
• When you add an attribute type, and each time you change an implementation defined segment, the tables are usually created during deployment. If, however, you do not have permissions to create tables in the database, you can use the option to generate the DDL. Refer to “Deploy Hub Configuration” on page 83 for more information.

For definitions of the fixed Initiate attribute types, refer to the IBM Initiate Master Data Service SDK Reference for Java and Web Services.

Note: Adding member attribute type functionality is intended for use by Initiate or partner implementers. MPI_seghead contains the name of the attribute type. MPI_segxfld describes the fields contained in MPI_seghead.

Procedure
1. In the Attribute Types view, select whether the new attribute is a Member Attribute Type or Relationship Attribute Type from the Type pull-down list.
2. Click Add beside the Attribute type pull-down list. A new attribute type is defined with a default name.
3. In the Properties view, type a Code and Table name for the attribute type.
4. In the Fields group box, click Add to add fields to the new attribute type. Complete steps a through e for each attribute type.
   a. In the Properties view, type the Name for the field. This is the field name that appears in the database.

   Note: There are standard fields that appear in all member attribute tables (i.e., memrecno, memseqno, caudrecno, maudrecno, recstat, attrrecno, asaidxno) that the Initiate software knows are used by all member attribute types. You do not have to add these fields for implementation-defined member attribute types.
   b. Type a Field Label. This is used by the APIs to programmatically access the field values.
   c. Select the Field data type from the pull-down list.
   d. Indicate whether this field is the table’s Primary Key, whether it is a Required field, and whether it is a Virtual field.
   e. Depending upon your selection of Field data type, the Field length field may not be visible.

   Note: If the attributes you are adding for this segment are new attributes to the Initiate software, you will need to add them via the Member Types:Attributes tab. Refer to “Member types” on page 89.
5. Once the attributes have been added, verify that the order displayed is the order in which you want them in the database. Use the Move up and Move down arrows to re-sort the order of attributes.
6. In most cases, the tables are created during deployment. If, however, you do not have permissions to create tables in the database, you can use the option to Generate the DDL. Refer to “Deploy Hub Configuration” on page 83 for more information.

Editing an attribute type
Most of the properties for an implementation-defined attribute type can be edited by following this process. Changes to pre-defined attribute types are not permitted.
Procedure
1. In the Attribute Types view, select Member Attribute Type or Relationship Attribute Type from the Type pull-down list.
2. Select the attribute type you want to edit from the Attribute type pull-down list. The Properties view refreshes to display the current properties.
3. Make changes to the properties for implementation-defined attribute types as needed.
4. Save the project.

Deleting an attribute type
Use caution when deleting attribute types, as once the project is saved, they cannot be restored by the use of Ctrl+Z or Edit > Undo after deletion.

Procedure
1. In the Attribute Types view, select Member Attribute Type or Relationship Attribute Type from the Type pull-down list.
2. Select the attribute type code from the Attribute type pull-down list and click Remove.

Results
Note: You can only delete implementation-defined member attribute types; pre-defined member attribute types cannot be deleted.

History
History allows you to store prior versions of the data so that you can reconstruct a view at a point in time in the past. Please note that the storage of History data is independent of nsExists or nsActive settings—these allow you to store multiple attribute versions for matching and linking. History data is not used for matching and linking, and the data is read-only. You can enable or disable History for any or all of the following configuration objects:
- Member Types
- Member Attribute Types
- Relationship Types
- Relationship Attribute Types

All types defined for each object are displayed in the list. The check box next to each item displays the current property setting; checked means that history is enabled.

An optional method to enable or disable history for individual configuration objects is to set the Uses history property on the tab used to manage its properties. For example, to enable history for the member attribute type MEMDATE, open the Attributes Types tab, select Member Attribute Types, select MEMDATE, and set its Uses history property to true in the Properties view.

Note: If you enable history for a member attribute type, but do not enable history for at least one member type that has an attribute whose type is the same, then enabling history for that attribute type will have no effect. This is true for relationship attribute type history and relationship type history as well. When this is the case, a warning will appear in the problems view indicating the condition. For example, a user enables history for the member attribute type MEMCONT, and
only enables history for the PERSON member type. Looking through the Attributes associated with the PERSON member type, we see that none of the attributes are of type MEMCONT. In this case, the following warning appears in the problems view:

Enabling history for member attribute type "MEMCONT" will have no effect because history has not been enabled for any applicable member types.

**Member type history**

To capture history for one or more member types, enable the checkbox for each object listed on the Member Types tab. You can use Select All or Deselect All to select or deselect all member types at once. The Uses history property for the member type will automatically be set to the same value. Refer to “Member types” on page 89. The Hub will store history only for the member types that are checked.

**Member attribute type history**

To capture history for one or more member attribute types, enable the checkbox for each object listed on the Member Attributes tab. You can click Select All or Deselect All to select or deselect all member attribute types at once. The Uses history property for the member attribute type will automatically be set to the same value. Refer to “Member Attribute Types” on page 107. The Hub will store history only for the member attribute types that are checked.

**Relationship type history**

To capture history for one or more relationship types, check or uncheck the checkbox for each object listed on the Relationship Types tab. You can click Select All or Deselect All to select or deselect all relationship types at once. The Uses history property for the relationship type will automatically be set to the same value. Refer to “Relationship types” on page 122. The Hub will store history only for the relationship types that are checked.

**Relationship attribute type history**

To capture history for one or more relationship attribute types, check or uncheck the checkbox for each object listed on the Relationship Attributes tab. You can use the Select All or Deselect All to select or deselect all relationship attribute types at once. The Uses history property for the relationship attribute type will automatically be set to the same value. Refer to “Relationship Attribute Types” on page 107. The Hub will store history only for the relationship attribute types that are checked.

**Informational sources**

Informational sources are thought of as “look up” data sources, typically providing “legal” values used to identify certain member attributes (e.g., state codes, credit card number/issuer or frequent customer number/issuer). The Initiate software does not send updates to an informational source.

**Informational source properties**

This section contains a list of the property names and descriptions that you can set for an informational source. Refer to this list when adding or editing an informational source.
Table 32. Informational source properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Entity Priority</td>
<td>Not currently used. Any value in this field is ignored.</td>
</tr>
<tr>
<td>Enable Review Identifier Task</td>
<td>Indicates whether records from this source will be compared for Review Identifier tasks.</td>
</tr>
<tr>
<td>Max length</td>
<td>The maximum length of identifiers for this source. Most sources are set at zero (0) to indicate no maximum.</td>
</tr>
<tr>
<td>Member Type</td>
<td>The member type to which this entity pertains. The default is the current member type, but the entity type can be set to All if you want it to pertain to all member types.</td>
</tr>
<tr>
<td>Min length</td>
<td>The minimum length of identifiers for this source. Most sources are set at zero (0) to indicate no minimum.</td>
</tr>
<tr>
<td>Physical code</td>
<td>A physical code provides a way to group sources, but does not affect Hub processing. For example, a hospital may have separate source systems for the laboratory, registration desk, and pharmacy. This code enables a grouping of these systems under one physical umbrella, yet retains a different source code identifier for each system.</td>
</tr>
<tr>
<td>Source code</td>
<td>This code is used as a prefix on all source-assigned record numbers from this source in the Initiate software (e.g., RMC:880189).</td>
</tr>
<tr>
<td>Source name</td>
<td>Name of this source.</td>
</tr>
</tbody>
</table>

Adding an informational source

During the initial configuration, it is likely that you will need to add one or more informational sources. Informational sources typically provide “legal” values that can be used as attribute values.

Procedure
1. In the Informational Sources view, click Add. A new informational source is added with default property values.
2. In the Properties view, set the appropriate values for each property. Refer to the table in Table 32 for property descriptions.
3. Save the project.

Editing an informational source

Most of the properties for an informational source can be edited by following this process.

Procedure
1. In the Informational Sources view, highlight the source you wish to change.
2. In the Properties view, make the necessary changes.
3. Save the project.

Deleting an informational source

Use caution when deleting informational sources, as once the project is saved, they cannot be restored by the use of Ctrl+Z or Edit > Undo after deletion.
Procedure
1. In the Informational Sources view, highlight the source you wish to delete.
2. Click Remove to the right of the Informational sources values table.
3. Save the project.

Tasks

There are six task types pre-configured in the Initiate software, four of which represent data issues and two that trigger a system action.

Data issue tasks are comprised of records which compare above the overlay threshold or between the clerical review and auto-link threshold levels and require manual review and resolution by users (e.g., IBM Initiate Inspector users). These task types are: Review Identifier, Potential Linkage, Potential Duplicate, and Potential Overlay.

System trigger tasks, HasShadow and PreMerge, are not seen in Initiate applications. Rather they are used primarily by the Message-Based Transaction Services to trigger an internal system action.

Additionally, the Initiate software manages categories or “Kinds” of tasks which affect how task records are updated and stored.

Member tasks
Those tasks where resolution updates affect a member. Potential Overlays, PreMerge, and HasShadow tasks are considered member-centric tasks.

Entity tasks
Represent tasks in which resolution work affects an entity:
- Potential Duplicate
- Potential Linkage
- Review Identifier

Relationship tasks
Along with Member Relationship and Entity Relationship, represent tasks associated with relationship types.

Custom tasks
Those created specifically for your implementation and populated by a Clover graph component or custom application via the API. When you create a new task type in IBM Initiate Workbench, its Kind is automatically set to Custom. More information on configuring and using custom tasks can be found in the IBM Initiate Master Data Service Feature Quick Start Guide. Also called Implementation defined tasks.

Use the Tasks tab to open the Tasks view and manage task types. The Tasks view is divided into two areas: Task Types and Task Statuses (workflows).

Task type properties

This section contains a list of the property names and descriptions that you can set for a task type. Refer to this list when adding or editing a task type.
Table 33. Task type properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Category is used primarily to group for reporting purposes and to act as a placeholder in the system. This field is editable.</td>
</tr>
<tr>
<td>Expiration days</td>
<td>For member, entity and custom kinds of tasks, this represents the number of days the task will be active before task expiration notification is triggered. Valid values are 0 to 32767. A value of 0 indicates that task expiration notification is disabled.</td>
</tr>
<tr>
<td>Is content</td>
<td>When the four data issue tasks (Potential Duplicate, Potential Linkage, Potential Overlay, and Review Identifier) are created, they are created based on the relationship of one record with another. System trigger tasks (HasShadow and PreMerge) are based on the content of a record. For example, if a record has at least one shadow attribute, a HasShadow task is created. Setting the Content flag to “true” instructs the software to create system trigger tasks. Again system trigger tasks are used primarily by the Message-Based Transaction Services and are not seen in end-user client applications. Thus, tasks seen in IBM Initiate Inspector have a “false” flag set in this field. This field is not editable.</td>
</tr>
<tr>
<td>Is linking</td>
<td>Indicates whether the task can result in a linkage of records. This field is read-only.</td>
</tr>
<tr>
<td>Label</td>
<td>The task label appears on client screens. This field is editable and can consist of 1 to 32 characters.</td>
</tr>
<tr>
<td>Task ID</td>
<td>A number representing the task. This field is read-only.</td>
</tr>
<tr>
<td>Task kind</td>
<td>Indicates how task records are updated and stored. This field is read-only.</td>
</tr>
<tr>
<td></td>
<td>• Member tasks are those tasks where resolution updates affect a member. Potential Overlays, PreMerge, and HasShadow tasks are considered member-centric tasks.</td>
</tr>
<tr>
<td></td>
<td>• Entity tasks—Potential Duplicate, Potential Linkage, and Review Identifier—represent tasks in which resolution work affects an entity.</td>
</tr>
<tr>
<td></td>
<td>• Relationship tasks, Member Relationship and Entity Relationship, represent tasks associated with relationship types.</td>
</tr>
<tr>
<td></td>
<td>• Custom tasks are those created specifically for your implementation. When you create a new task type in Initiate IBM Initiate Workbench, its Kind is automatically set to Custom. (Note that the Task kind field in the Properties view shows I, which stands for Implementation defined.)</td>
</tr>
<tr>
<td>Task type</td>
<td>Short text description of the task type.</td>
</tr>
</tbody>
</table>

Task status properties

This section contains a list of the property names and descriptions that you can set for a task status. Refer to this list when adding or editing a task status. Task status information relates to workflow and is stored in MPI_tskstat.

Table 34. Task status properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Category is used primarily to group for reporting purposes and to act as a placeholder in the system.</td>
</tr>
<tr>
<td>Is deletable</td>
<td>Indicates whether a task with this status can be deleted.</td>
</tr>
</tbody>
</table>
Adding a task type

Task types you add use the Custom task kind and can be deleted. System-defined tasks cannot be added or deleted.

Procedure

1. In the Tasks view, click Add in the Task Types group box.
2. In the Properties view, set the appropriate values for each property.

   Note: Some fields are read-only and cannot be edited.
3. Save the project.

   Results

Refer to Table 33 on page 114 for property descriptions.

Editing a task type

Some of the properties for a task type can be edited by following this process. Several fields are read-only and cannot be edited.

About this task

Refer to Table 33 on page 114 for property descriptions.

Procedure

1. In the Task Types area, highlight the desired task type.
2. In the Properties view, make changes as necessary.
3. Save the project.

Deleting a task type

Predefined tasks cannot be removed; only custom tasks can be deleted. Use caution when deleting task types, as once the project is saved, they cannot be restored by the use of Ctrl+Z or Edit > Undo after deletion.

Procedure

1. In the Task Types area, highlight the undesired type.
2. Click Remove.
Adding a task status

Add a task status to enable users to designate the status of an existing task.

Procedure
1. In the Tasks view, click Add in the Task Statuses group box.
2. In the Properties view, set the appropriate values for each property. Refer to Table 34 on page 114 for property descriptions.
   
   **Note**: The Status ID field is read-only and cannot be edited.
3. Save the project.

Editing a task status

Most of the properties for a task status can be edited by following this process.

About this task

These properties are editable except where noted in Table 34 on page 114.

**Note**: You cannot edit system task statuses (OB-sent or OB-reject) from this application; these can only be edited via the dicboot file.

Procedure
1. In the Task Status area, highlight the desired task status.
2. In the Properties view, make changes as necessary.
3. Save the project.

Deleting a task status

Use caution when deleting task statuses, as once the project is saved, they cannot be restored by the use of Ctrl+Z or Edit > Undo after deletion. If the possibility exists that you may use a task status at a later date, marking it with an Inactive status rather than deleting is suggested.

About this task

**Note**: Unexamined, Resolved, Deferred, OB-Sent, and OB-Reject statuses must be present for the Hub processes to operate correctly.

Procedure
1. In the Task Status area, highlight the undesired status.
2. Click Remove.

Linkages

Linkages are formed either automatically for records scoring above the auto-link threshold or manually by users during task resolution. The purpose of linkages is to enable an accurate enterprise-wide view of a member. Use this functionality to add or edit linkage types and associated statuses.

The Linkages view is divided into two areas: Linkage types which define valid entity relationships, and Linkage statuses, which represent the workflow status of the enterprise relationships.
Linkage type properties

This section contains a list of the property names and descriptions that you can set for a linkage type. Refer to this list when adding or editing a linkage type.

Table 35. Linkage type properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Category is used primarily to group for reporting purposes and to act as a placeholder in the system. This field is editable.</td>
</tr>
<tr>
<td>Label</td>
<td>A short description of the status. This field is editable.</td>
</tr>
<tr>
<td>Linkage ID</td>
<td>System-assigned number used internally by the software. This field is read-only.</td>
</tr>
<tr>
<td>Linkage kind</td>
<td>Identifies whether the linkage is member-centric (linkage resolution affects a member), entity-centric (linkage resolution affects an entity), or both member- and entity-centric. This field is read-only.</td>
</tr>
</tbody>
</table>
Table 35. Linkage type properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linkage type</td>
<td>The specific type of enterprise relationship. This entry appears on client screens. There are twelve defined linkage types used by the Initiate software, all of which must exist for Hub processing. This field is read-only.</td>
</tr>
<tr>
<td>AutoLink-MS</td>
<td>Records compared from multiple sources above the auto-link threshold, and a common Enterprise ID was assigned by the Initiate software.</td>
</tr>
<tr>
<td>AutoLink-SS</td>
<td>Records compared from the same source above the auto-link threshold, and a common Enterprise ID was assigned by the Initiate software.</td>
</tr>
<tr>
<td>AutoUnlink</td>
<td>A previously auto-linked record that, because of new information, no longer compares below the auto-link threshold with another record and has been automatically unlinked by the Initiate software. Once unlinked, if the record compares above the clerical review threshold, it may be placed into a task.</td>
</tr>
<tr>
<td>Delete</td>
<td>An IBM Initiate Inspector user logically deleted the record so that the Initiate software no longer includes it in the algorithmic comparison.</td>
</tr>
<tr>
<td>ManuLink-MS</td>
<td>An IBM Initiate Inspector user manually linked records from multiple sources.</td>
</tr>
<tr>
<td>ManuLink-SS</td>
<td>An IBM Initiate Inspector user manually linked records from the same source.</td>
</tr>
<tr>
<td>ManuUnlink</td>
<td>An IBM Initiate Inspector user manually unlinked records that had previously shared a common Enterprise ID.</td>
</tr>
<tr>
<td>Merge</td>
<td>Indicates when Master Data Engine software received a merge message from the source system.</td>
</tr>
<tr>
<td>NoChange</td>
<td>An IBM Initiate Inspector user updates the workflow status, but does not make a change to the Enterprise ID.</td>
</tr>
<tr>
<td>Premerge</td>
<td>An IBM Initiate Inspector user resolved a record to be merged and set the Surviving Enterprise ID and Surviving Source ID. When Master Data Engine software receives the merge message from the source, this record is set to “Merge.”</td>
</tr>
<tr>
<td>Undelete</td>
<td>An IBM Initiate Inspector user activated the record after it had been deleted.</td>
</tr>
<tr>
<td>UnMerge</td>
<td>A record which was previously merged that has been unmerged.</td>
</tr>
</tbody>
</table>
Linkage status properties

This section contains a list of the property names and descriptions that you can set for a linkage status. Refer to this list when adding or editing a linkage status.

Table 36. Linkage status properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Category is used primarily to group for reporting purposes and to act as a placeholder in the system.</td>
</tr>
<tr>
<td>Is deletable?</td>
<td>Indicates if records with this status can be deleted.</td>
</tr>
<tr>
<td>Is resolved?</td>
<td>Indicates if linkages assigned this status result in task resolution. If you are using the outbound broker, Is resolved needs to be set to system.</td>
</tr>
<tr>
<td>Is updatable?</td>
<td>Indicates whether records with this status can be updated.</td>
</tr>
<tr>
<td>Label</td>
<td>A short description of the status.</td>
</tr>
<tr>
<td>Linkage ID</td>
<td>System-assigned number used internally by the software.</td>
</tr>
<tr>
<td>Linkage status</td>
<td>Linkage status identifies the state of the linkage (records) in the workflow process. There are three defined linkage statuses which are pre-configured in the Initiate software.</td>
</tr>
<tr>
<td>Unexamined</td>
<td>The linkage has not been reviewed. The Unexamined status is required to exist for Hub processing.</td>
</tr>
<tr>
<td>Examined-OK</td>
<td>The linkage has been reviewed and confirmed appropriate by a user. (Not required.)</td>
</tr>
<tr>
<td>Examined-ERROR</td>
<td>The linkage has been reviewed and confirmed as incorrect. (Not required.)</td>
</tr>
</tbody>
</table>

Editing a linkage type

Some of the properties for a linkage type can be edited by following this process.

About this task

Linkage types cannot be added or removed. Properties are read-only except where noted in Table 35 on page 117

Procedure

1. In the Linkage types group box, highlight the linkage type to be edited.
2. In the Properties view, make the necessary changes to the desired properties.
3. Save the project.

Adding a linkage status

The linkage status represents the workflow status of the enterprise relationships.

Procedure

1. In the Linkage statuses group box, click Add. A new linkage status is created with default properties.
2. In the Properties view, set the appropriate values for each property. Refer to the table above for property descriptions.
3. Save the project.

**Editing a linkage status**

Most of the properties for a linkage status can be edited by following this process.

**About this task**

Properties are read-only except where noted in Table 36 on page 119.

**Procedure**

1. In the Linkage statuses group box, highlight the linkage status to be edited.
2. In the Properties view, make the necessary changes to the desired properties.
3. Save the project.

**Deleting a linkage status**

Use caution when deleting linkage statuses, as once the project is saved, they cannot be restored by the use of Ctrl+Z or Edit > Undo after deletion.

**About this task**

Note: The Unexamined status is required for normal processing by the Hub, and should not be deleted.

**Procedure**

1. In the Linkage statuses group box, highlight the undesired linkage status.
2. Click Remove.
3. Save the project.

---

**Events**

If database interaction requests are event-oriented, the event must be defined in the data dictionary. Events not listed in MPLEvttype and configured in the Message-Based Transaction Services are rejected by the Initiate software. (Refer to the IBM Initiate Master Data Service Message Broker Suite Reference for configuration information.)

**Event properties**

This section contains a list of the property names and descriptions that you can set for an event. Refer to this list when adding or editing an event.

*Table 37. Event properties*

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Category is used primarily to group for reporting purposes and to act as a placeholder in the system.</td>
</tr>
<tr>
<td>Event ID</td>
<td>System-assigned number used internally by the software. Event ID is read-only.</td>
</tr>
<tr>
<td>Event type</td>
<td>Defines the event type. Note that there are no reserved values used by the Initiate software. This setting is most often a precise string as defined by an organization's data transactions. For example, the above example shows HL7 event types.</td>
</tr>
<tr>
<td>Label</td>
<td>A short string labeling the event.</td>
</tr>
</tbody>
</table>
Adding an event

Creating event types is the first step in enabling the system to process event-oriented database interaction requests. The event types must then be configured in the Message-Based Transaction Services.

Procedure
1. In the Events view, click Add. A new event type appears in the list of event types with default properties.
2. In the Properties view, set the appropriate values for each property. Refer to the table in “Events” on page 120 for property descriptions.
3. Save the project.

Results

Refer to the IBM Initiate Master Data Service Message Broker Suite Reference for configuration information.

Editing an event

Most of the properties for an event can be edited by following this process.

About this task

Properties are read-only except where noted in Table 37 on page 120

Procedure
1. In the Events view, highlight the event type you want to edit.
2. In the Properties view, edit the properties as desired.
3. Save the project.

Deleting an event

Use caution when deleting event types, as once the project is saved, they cannot be restored by the use of Ctrl+Z or Edit > Undo after deletion.

About this task

Note: Deleting an event type does not affect Hub processing, but it does affect Message-Based Transaction Services. If you delete an event type and do not update the Message-Based Transaction Service parameters, messages with this event type are rejected.

Procedure
1. In the Events view, highlight the event type you want to delete.
2. Click Remove.
3. Save the project.
Relationship types

A Relationship Type is a type of association that can exist between two different (or same) entity types. For example, a person can manage another person, or an organization can legally own another organization. Or a provider can provide health services to a patient. Relationship types are used in IBM Initiate Inspector.

Relationship type properties

This section contains a list of the property names and descriptions that you can set for a relationship type. Refer to this list when adding or editing a relationship type.

Table 38. Relationship type properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directed</td>
<td>Indicates whether or not there is a start and end point for the relationship.</td>
</tr>
<tr>
<td>Enable tasks</td>
<td>Indicates whether the relationship type can make use of tasks. Setting this property to false prevents tasks of kind Entity Relationship or Member Relationship from being created for this relationship type. The default setting is true.</td>
</tr>
<tr>
<td>Type</td>
<td>A simple descriptor for the relationship. This value must be unique.</td>
</tr>
<tr>
<td>Left Enttype (Entity 1)</td>
<td>Indicates the entrencno of the item on the left side of the relationship. When an entity type is selected here, the Value column for the following nineteen properties (shown in the Properties view when the Left Enttype is expanded) is automatically populated with the selected entity's matching properties.</td>
</tr>
<tr>
<td>Linker can delete linker-created relationships?</td>
<td>Indicates whether the relationship linker can delete relationships it created if they do not conform to the established rules.</td>
</tr>
<tr>
<td>Linker can delete user-created relationships?</td>
<td>Indicates whether the relationship linker can delete relationships created by a user if they do not conform to the established rules.</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>Indicates the constraints on a relationship. Choices are 1 to 1, 1 to Many and Many to Many.</td>
</tr>
<tr>
<td>Relationship Name</td>
<td>The consumable description of the relationship type.</td>
</tr>
<tr>
<td>Relationship Type</td>
<td>A short description of the relationship, usually a single word.</td>
</tr>
<tr>
<td>Reltypeno</td>
<td>A read-only field containing the RELTYPENO field from the Hub database.</td>
</tr>
<tr>
<td>Required Hierarchy</td>
<td>Indicates whether this entity is required to appear at least once as the left or right entity in a relationship type. If Required Hierarchy is true, then the entity type for both left and right entities must be the same. Required Left and Required Right must both be false if Required Hierarchy is true.</td>
</tr>
</tbody>
</table>
Table 38. Relationship type properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required Left</td>
<td>Indicates whether the entity (of type right) must appear at least once on the RIGHT side of a relationship. If Directed is false and Right Enttype = Left Enttype, Required Left must be false. For example, a 1-to-Many relationship exists where the left entity is the boss and the right entity is the employee. When Required Left is true for this relationship, an employee must have a boss, so the employee must appear on the right side of a relationship at least once.</td>
</tr>
<tr>
<td>Required Right</td>
<td>Indicates whether the entity (of type left) must appear at least once on the LEFT side of a relationship. If Directed is false and Right Enttype = Left Enttype, Required Right must be false. For example, a 1-to-Many relationship exists where the left entity is the boss and the right entity is the employee. When Required Right is true for this relationship, a boss must have at least one employee, so the boss must appear on the left side of a relationship at least once.</td>
</tr>
<tr>
<td>Right Enttype (Entity 2)</td>
<td>Indicates the entrecno of the item on the right side of the relationship. When an entity type is selected here, the Value column for the following nineteen properties (shown in the Properties view when the Right Enttype is expanded) is automatically populated with the selected entity’s matching properties.</td>
</tr>
<tr>
<td>Uses common membership rule</td>
<td>Indicates whether links are determined by common members in the entities involved in the relationship (implicit).</td>
</tr>
<tr>
<td>Uses history</td>
<td>Indicates whether history will be kept for the selected member type. This property can also be set on the History tab. (See “History” on page 110.)</td>
</tr>
</tbody>
</table>

Relationship type attributes properties

This section contains a list of the property names and descriptions that you can set for a relationship type attribute. Refer to this list when adding or editing a relationship type attribute.

Table 39. Relationship type attributes properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attrrecno</td>
<td>A read-only field containing the ATTRRECNO field from the Hub database.</td>
</tr>
<tr>
<td>Code</td>
<td>The code used to define the attribute.</td>
</tr>
<tr>
<td>Description</td>
<td>The description of the attribute.</td>
</tr>
<tr>
<td>Label</td>
<td>A label for the attribute. This label appears under the Attribute Name column when you Inspect Relationships in IBM Initiate Inspector.</td>
</tr>
<tr>
<td>Maximum active values</td>
<td>The number in this field controls status assignment for updated values. For example, if this number is set to 1 or 0, only one attribute value of this type will have an active status. Any updates to that value results in the new value being active and the previous value becoming inactive.</td>
</tr>
</tbody>
</table>
Table 39. Relationship type attributes properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum existing values</td>
<td>This number indicates how many updated values are stored in history. Whatever number this is set to, the software stores that number of values—regardless of status—in history. For example, if it is set to 2 and there are two attribute values currently stored, a third value causes the earliest value to be removed (trimmed) from the database.</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the attribute.</td>
</tr>
<tr>
<td>Type</td>
<td>The Attribute Type associated with the attribute. These are defined on the Attribute Types view for Relationship attribute types.</td>
</tr>
</tbody>
</table>

Relationship type rules properties

This section contains a list of the property names and descriptions that you can set for a relationship type rule. Refer to this list when adding or editing a relationship type rule.

Table 40. Relationship type rules properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left is reference/foreign key</td>
<td>When set to true, this property will signal the Hub to generate an &quot;Invalid References&quot; error if a member attribute used as the foreign key (FK) in a rule has a value that does not match another member attribute used as the primary key (PK) in the rule.</td>
</tr>
<tr>
<td>Right is reference/foreign key</td>
<td>When set to true, this property will signal the Hub to generate an &quot;Invalid References&quot; error if a member attribute used as the foreign key (FK) in a rule has a value that does not match another member attribute used as the primary key (PK) in the rule.</td>
</tr>
</tbody>
</table>

Adding relationship types

To create an association between two entity types, add a new relationship type.

Procedure

1. Click the Relationship Types tab.
2. Click Add. A new relationship type property is created with default values.
3. In the Properties view, set the appropriate values for each property.
4. Add attributes and rules as needed, following the instructions later in this chapter.
5. Save the project.

Editing relationship types

Most of the properties for a relationship type can be edited by following this process. To edit an association between two entity types, edit an existing relationship type.
Procedure
1. Click the Relationship Types tab.
2. Select the desired relationship type.
3. In the Properties view, make any necessary changes to the desired properties.
4. Save the project.

Deleting relationship types
Use caution when deleting relationship types, as once the project is saved, they cannot be restored by the use of Ctrl+Z or Edit > Undo after deletion.

Procedure
1. Click the Relationship Types tab.
2. Select the unwanted relationship type.
3. Click Remove.
4. Save the project.

Adding relationship type attributes
Adding Relationship type attributes enables you to define attributes specific to a relationship type.

Procedure
1. In the Relationship Types view, highlight the desired relationship type.
2. Click Add on the Attributes sub-view. A new relationship type attribute property is created with default values.
3. In the Properties view, set the appropriate values for each property. In order to select a value for the Type property, an Attribute type of type Relationship attribute type must have been predefined. Refer to "Relationship Attribute Types" on page 107 for more information.
4. Save the project.

Editing relationship type attributes
Most of the properties for a relationship type attribute can be edited by following this process.

Procedure
1. In the Relationship Types view, highlight the desired relationship type.
2. On the Attributes tab, select the attribute to change.
3. In the Properties view, make any necessary changes to the desired properties.
4. Save the project.

Deleting relationship type attributes
Use caution when deleting relationship type attributes, as once the project is saved, they cannot be restored by the use of Ctrl+Z or Edit > Undo after deletion.

Procedure
1. In the Relationship Types view, highlight the desired relationship type.
2. On the Attributes tab, select the unwanted attribute.
3. Click Remove in the Attributes sub-view.
4. Save the project.
Adding relationship type rules

The hub can draw lines and use attributes of entities on both sides to derive relationships and create them. Relationship type creation rules are based on the concept of source:attribute:field of the left entity equaling the source:attribute:field of the right entity.

About this task

Note: The “attribute” used in relationship type rules is a Member Attribute Type configured for the selected source, and not the attribute for the relationship type.

When a relationship type rule is added, a hidden Query Role is added to the algorithm, requiring that the data be re-derived. In this case, be sure to run the required jobs after the configuration is uploaded to the Hub. See “Link Relationships” on page 71 for more information.

Procedure

1. In the Relationship Types view, highlight the desired relationship type.
2. Click Add on the Rules sub-view. A new relationship type rule property is created with default values.
3. In the Rules sub-view, set the appropriate values for each property.
4. Click Move up or Move down to order the rules as desired.
5. Save the project.

Editing relationship type rules

Most of the properties for a relationship type rule can be edited by following this process.

Procedure

1. In the Relationship Types view, highlight the desired relationship type.
2. On the Rules tab, select the rule to change.
3. In the Rules sub-view, make any necessary changes to the desired properties.
4. Save the project.

Deleting relationship type rules

Use caution when deleting relationship type rules, as once the project is saved, they cannot be restored by the use of Ctrl+Z or Edit > Undo after deletion.

Procedure

1. In the Relationship Types view, highlight the desired relationship type.
2. On the Rules tab, select the unwanted rule.
3. Click Remove in the Rules sub-view.
4. Save the project.

Applications

Use the functions in this component to mark an application active or inactive, and to manage MPI_appprop table entries.
To access the Applications component, select the Applications tab. From the Applications view, you can add and remove Initiate applications implemented at your site, as well as adjust their properties.

**Application properties**

This section contains a list of the property names and descriptions that you can set for an application. Refer to this list when adding or editing an application.

*Table 41. Application properties*

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application ID</td>
<td>The APPRECNO for the application. This field is read-only.</td>
</tr>
<tr>
<td>Application name</td>
<td>The name of the application.</td>
</tr>
</tbody>
</table>

**Adding an application**

In previous versions of Initiate software, applications that were used to access the Hub data were added using this process. The most current applications were not added using this view.

**Procedure**

1. Select the Applications tab.
2. Click Add beside the Application pull-down list. A new application is created with a default name.
3. In the Properties view, set the Application name.
4. Save the project.

**Results**

The application now appears in the pull-down list.

**Editing an application**

The application name property for an application can be edited by following this process.

**Procedure**

1. Select the Applications tab.
2. Select the application you want to edit from the Application pull-down list.
3. In the Properties view, edit the Application name as desired.
4. Save the project.

**Deleting an application**

Use caution when deleting applications, as once the project is saved, they cannot be restored by the use of Ctrl+Z or Edit > Undo after deletion.

**Procedure**

1. Select the Applications tab.
2. Select the application you want to delete from the Application pull-down list.
3. Click Remove beside the Application pull-down list.
Application properties

In previous versions of Initiate software, the display order of attributes, as well as the look of search screens and views in applications was handled by directly editing the MPI_appprop table. By using the Add and Remove functions, you can more easily edit the table. For more detailed information about adding validation properties to Initiate IBM Initiate Workbench, consult Appendix B, “Algorithm validation rule syntax,” on page 335.

Note: Initial screen configuration is performed by the Initiate Project Team or Partner during the implementation cycle.

Table 42. Application Properties properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A description of the property.</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the property. Some examples of property names are:</td>
</tr>
<tr>
<td></td>
<td>• ...WIDTH – width of a screen or column</td>
</tr>
<tr>
<td></td>
<td>• ...CTRL – controls/fields/columns that display on a screen, e.g., fields on the Task Summary view in IBM Initiate Inspector (TASKSUMCOL)</td>
</tr>
<tr>
<td></td>
<td>• ...CTGCOL – defines attribute columns to display</td>
</tr>
<tr>
<td></td>
<td>• ...ATTRDISPLAYORDER – the order in which attributes display</td>
</tr>
<tr>
<td>Primary index</td>
<td>An application-specific number that defines attribute grouping; i.e., ensures that attributes which should be grouped by index number (asaiidxno in MPI_audatr) get the same value. This number must be a unique value across application properties (no two application properties should share a Primary Index), and it is a good practice to number it sequentially.</td>
</tr>
<tr>
<td>Secondary index</td>
<td>This field is an application specific field (stored in the recno field) and defines the attribute sort order on a display. This number must be a unique value across application properties (no two application properties should share a Secondary Index), and it is a good practice to number it sequentially.</td>
</tr>
<tr>
<td>Value</td>
<td>The value of the property. For screen settings, the value determines what attributes display on a given application screen. For example on a search screen, settings in this table determine what attributes you can search on and the order in which the attributes display on the screen. You will also see settings for such items as timeout. This entry is saved in the propval field in MPI_appprop.</td>
</tr>
</tbody>
</table>

Adding application properties

About this task

To add new application properties for an existing application:

Procedure

1. Click the Applications tab.
2. Select the desired application from the Application pull-down list.
3. Click Add within the Application Properties group box. A new application property is created with default values.
4. In the Properties view, set the appropriate values for each property.
5. Save the project.

**Editing application properties**

Most of the properties for an application property can be edited by following this process.

**Procedure**

1. Click the Applications tab.
2. Select the desired application from the Application pull-down list.
3. In the Properties view, make any necessary changes to the desired properties.
4. Save the project.

**Deleting application properties**

**About this task**

Use caution when deleting application properties, as once the project is saved, they cannot be restored by the use of Ctrl+Z or **Edit > Undo** after deletion.

**Procedure**

1. Click the Applications tab.
2. Select the desired application from the Application pull-down list.
3. Click **Remove** within the Application Properties group box.
4. Save the project.

**Initiate Web Reports properties**

The following properties are specific to the Initiate Web Reports application. When new reports become available, they are added to the properties list.

*Table 43. Web Reports properties*

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>reports.auditor</td>
<td>auditorDuplicationSummaryStatistics</td>
</tr>
<tr>
<td>reports.auditor</td>
<td>auditorEventActivity</td>
</tr>
<tr>
<td>reports.auditor</td>
<td>auditorEventDetail</td>
</tr>
<tr>
<td>reports.auditor</td>
<td>auditorEventSummary</td>
</tr>
<tr>
<td>reports.auditor</td>
<td>auditorLinkageManagementOverview</td>
</tr>
<tr>
<td>reports.auditor</td>
<td>auditorNotesDetail</td>
</tr>
<tr>
<td>reports.auditor</td>
<td>auditorPendingResolutionDetail</td>
</tr>
<tr>
<td>reports.auditor</td>
<td>auditorRecordResolutionDetail</td>
</tr>
<tr>
<td>reports.auditor</td>
<td>auditorTaskCreationDetail</td>
</tr>
<tr>
<td>reports.auditor</td>
<td>auditorTaskManagementDetail</td>
</tr>
<tr>
<td>reports.auditor</td>
<td>auditorTaskManagementOverview</td>
</tr>
<tr>
<td>reports.hubmgr</td>
<td>hubmgrInitiateAdditions</td>
</tr>
<tr>
<td>reports.hubmgr</td>
<td>hubmgrUserActivitySummary</td>
</tr>
<tr>
<td>reports.operational</td>
<td>operationalOutstandingTasksBySource</td>
</tr>
<tr>
<td>reports.operational</td>
<td>operationalTaskAssignmentByOwner</td>
</tr>
</tbody>
</table>
Table 43. Web Reports properties (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>reports.operational</td>
<td>operationalTaskCountByTagType</td>
</tr>
</tbody>
</table>

**Adding Initiate Web Reports**

**About this task**

If a new report becomes available, you can add its configuration information to this table.

**Procedure**

1. Click the Applications tab.
2. Select **Identity Hub Reports** from the Application pull-down list.
3. For each report you wish to enable, click **Add** beside the Application Properties group box. A new application property is created with default values.
4. Use the Names and Values for the reports you wish to enable, as described in Table 43 on page 129.
5. Add a Primary Index and Secondary Index for each report.
6. Optional: The Record Status is set to Active by default. If you wish to turn a report off globally, set the Record Status to Inactive in the Properties view.
7. Repeat steps 3-6 as needed.
8. Save the project.

**Handlers**

You can register custom callout handlers on the Handlers tab. More detailed information on using this tab is located in Chapter 8, “Callout handlers,” on page 199.

**EDT (Enumerated Data Type)**

An EDT (enumerated data type) is one in which incoming data has a finite, known standard expression. For example: Sex – M/F; marital status – married, single; or state – AZ, CA, FL, and so forth. EDT elements are the standard values that an EDT recognizes.

There are multiple uses of EDTs; one is providing known selection values in clients. For example, an EDT called “SEX” is defined in MPI_edthead with elements of “M” and “F” defined in MPI_edtelem. The elements have expansion values (Descriptions) of “Male” and “Female.” A screen control which points to an EDT code would return a selection list for SEX with the options of Male and Female. Refer to the IBM Initiate Master Data Service Data Model Description (MPI_edthead and MPI_edtelem) for further information on how EDTs are used.

**Enumerated data type properties**

This section contains a list of the property names and descriptions that you can set for an EDT. Refer to this list when adding or editing an EDT.
Table 44. Enumerated Data Type properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type code</td>
<td>The EDT identifier.</td>
</tr>
<tr>
<td>Data type name</td>
<td>The EDT value. This entry appears on client screens and selection lists.</td>
</tr>
<tr>
<td>Data type owner</td>
<td>Owner provides a way to categorize code types for security.</td>
</tr>
</tbody>
</table>

Adding an EDT code

About this task

Each environment uses various system-specific EDTs.

Procedure

1. In the Enumerated Data Type view, beside the pull-down list of existing EDTs, click Add. A new EDT is created with no members. Its default name is EnumTypeX, where X represents a unique number.
2. In the Properties view, set the appropriate values for each property. Refer to the table above for property descriptions.
3. Save the project.

Editing an EDT code

All of the properties for an Enumerated Data Type code can be edited by following this process.

Procedure

1. In the Enumerated Data Type view, select the code you want to edit from the Data types pull-down list.
2. In the Properties view, make the necessary changes.
3. Save the project.

Deleting an EDT code

Procedure

1. In the Enumerated Data Type view, select the code you want to delete from the Data types pull-down list.
2. Click Remove. You do not need to delete the elements first.
3. Save the project.

Sorting EDT elements

About this task

Click Sort values to sort the EDT elements by the Value column.

Adding an EDT element

About this task

To add elements from an existing EDT, first select the EDT from the Data types pull-down list in the Enumerated Data Type view.
**Procedure**
1. To add elements, click Add to the right of the Data type values table. A new element is created with default values.
2. In the Properties view or in the Data type values table, set the Value and Description for each entry.
3. Save the project.

**Editing an EDT element**
All of the properties for an Enumerated Data Type element can be edited by following this process.

**Procedure**
1. To edit elements from an existing EDT, first select the EDT from the Data types pull-down list in the Enumerated Data Type view.
2. In the Data type values table, highlight the element you wish to change.
3. In the Properties view, make the necessary changes to the properties.
4. Save the project.

**Deleting an EDT element**

**About this task**
To delete elements from an existing EDT, first select the EDT from the Data types pull-down list in the Enumerated Data Type view.

**Procedure**
1. In the Data type values table, highlight the element you want to delete.
2. Click Remove to the right of the Data type values table.
3. Save the project.

**Strings**
Strings enable you to create “rules” or “guidelines” that instruct the algorithm on how to handle certain incoming data values. For example, if entering member data when not all of the member information is available, “dummy” data may be entered in some required fields (such as 999999999 for a Social Security number or 5555555555 for a phone number). If this same dummy data were entered for every member in which the Social Security number was not available, you could end up with multiple records having the same 999999999 number and thus, causing inaccurate comparison scores. By defining these values in an anonymous string, the algorithm knows to ignore these values during bucketing and comparison.

Another example of string use is in mapping specialty codes or professional titles that can be part of a person or business name. Standardization routines are used to standardize member names (first, last, middle); however these names can often include additional words or titles that can assist in achieving more accurate comparison and identification. Abstracts are a means of providing this additional information when comparing and identifying members. In order to use the abstracts in bucketing and comparisons, the values must be standardized. The result of this is the “abstract value.” Abstracts are defined codes that map to specific words found in member data which represent the same thing. The words “Heart” and “Cardio,” while not the same, may represent the same specialty.
Abstracts may be inferred from two places: 1) from words embedded in the member's name, or 2) from specific attribute values. For example, a comparison of the following records:

“J Smith” with a specialty code attribute of “Cardiologist”

“J Smith Heart Center”

“J Smith” with a specialty code attribute of “Dentist”

“Cardiologist” and “Heart” both map to an abstract value of “CRD,” whereas “Dentist” maps to an abstract of “DNT.” The abstract comparison between “J Smith, Cardiologist” and “J Smith Heart Center” adds a small amount to the score between these members to distinguish it from comparisons with “J Smith, Dentist.”

**String properties**

This section contains a list of the property names and descriptions that you can set for a string. Refer to this list when adding or editing a string.

**Table 45. String properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>The string type identifier.</td>
</tr>
<tr>
<td>Description</td>
<td>Brief description of the string.</td>
</tr>
<tr>
<td>Name</td>
<td>The string name.</td>
</tr>
<tr>
<td>String Value File</td>
<td>The file name containing the string values.</td>
</tr>
<tr>
<td>Type</td>
<td>Indicates the style of transformation occurring for this string. For example, ANON means this string is for anonymous values. This field also indicates the database table in which the values for this string are stored; e.g., ANON = MPI_stranon.</td>
</tr>
</tbody>
</table>

**Anonymous addresses**

The Hub handles anonymous addresses differently than other types of anonymous attributes. If it finds the ADDR2-MANON string code in the mpi_strhead table, it checks for the anonymous address values in the mpi_stranon table. Furthermore, the algorithm’s xxADDR2 standardization functions do not require an “Anonymous string code” to be assigned, which is the usual trigger for the engine to incorporate other types of anonymous attributes, such as names.

An anonymous address String Code’s Code value is ADDR2-MANON (see “Adding a string” on page 137), and its string value file (see “Adding string code values” on page 138) must contain standardized addresses. One method of obtaining standardized addresses is described in “Standardizing anonymous address strings” on page 141.

**String Examples**

Some examples of strings and their uses are described below. Note that this is only an example and does not contain all possible configurations.
**ADDR-TOK**

Address tokens are used to standardize portions of a member address for comparison and bucketing. Because contents of addresses can vary (e.g., a single address can be entered as: 124 W. Park Dr. or 124 West Park Drive), standardizing certain elements enables a common format to be used during comparison. For example, “apartment” may be formatted as “APT, or “Drive” to “DR”. Address tokens are defined in mpi_strword.

**Note:** There is a difference between tokens and abstracts. Tokens are items extracted from a “unit” of information, such as the word “Drive” in an address. Abstracts are items from which we derive a meaning, such as “DDS” meaning “Dentist.”

**ADDR2-MANON**

Occasionally when creating a member record, a street address may not be available. A registrar may enter “1234 Main Street, Anytown, NY 12345” as the address. Providing a list of anonymous address values enables Master Data Engine software to ignore the value and not assign a score. To do this, use string code whose Code value is ADDR2-MANON. Anonymous address values are defined by adding the anonymous addresses to the string value file associated with the ADDR2-MANON string code. Values are stored in the mpi_stranon table. *Note that anonymous addresses must be entered in standardized format.*

**ANAME**

The Anonymous Name string code with aname.txt string value file is used when creating a member record for which certain name information is not available. For example when creating a member record for a newborn, a registrar may enter “Babyboy” as the first name. Providing a list of anonymous name values enables Master Data Engine software to ignore the value and not assign a score. Anonymous name values are defined in the mpi_stranon table.

**BXNM-ABS**

The BXNM-ABS string code is used to define and manage the use of abstracts for specialty codes in the Master Data Engine software. During a name standardization process, each word/token making up a member or business name is examined to see if a “mapping” exists to an abstract value. If so, the abstract value is added to the comparison data. The mappings for a name token to abstracts are defined in the mpi_strequi table.

Mappings for attributes to abstracts in MPI_strequi are slightly different than other mappings. Since attribute values across multiple sources may not be the same, the MPI_strequi.strcode is used to define whether the mapping should apply across all sources or only to a specific source. The mpi_strequi records have the following formats:

For global mappings across all sources:

\[
\text{strcode} = \text{SA0-ABS}; \text{strval1} = \text{abstract value}; \text{strval2} = \text{attribute value}.
\]

For source specific mappings:

\[
\text{strcode} = \text{SA<srcrecno>-ABS}; \text{strval1} = \text{abstract value}; \text{strval2} = \text{attribute value}.
\]
For example, a specialty code attribute value “PST” might map to an abstract value of “RHB” for Source 13, whereas in Source 14 “5C” might map to “RHB.” To accomplish this, define an MPI_strequi record, strcode=SA13-ABS, to map “PST” to the abstract “RHB” and another record with strcode=SA14-ABS to map “5C” to “RHB.”

**Note:** Each mapping must be defined individually.

**BXNM-TOK**

This string value is used to standardize certain components of a business name. BXNM-TOK values are stored in mpi_strword. Remember that tokens are items extracted from a unit of information.

**BYR**

Some implementations, for example those using the Initiate Offender algorithm, utilize comparisons on a member’s birth year (without day and month). This string enables you to define an incoming year value and the converted value to use in bucketing and comparison. Values are stored in mpi_wgtval.

**DATE**

This anonymous values string is used when you want certain dates ignored during comparison. For example if a birth date is not available when creating a member record, a “dummy” value may be entered. This can also be used if you wanted to prevent all birth dates prior to a specific date from being used in comparison. Values for this string are stored in mpi_stranon.

**HGT**

This string is used to generate a standardized member height value (range of heights) for bucketing and comparison. Often the height entered for a member is only an approximation. Using the HGT string, you can define height ranges. For example, you might specify that all height values of 4’8” to 5’2” should have a bucket value of 5’0” and 53” to 57 should have a bucket value of 55”. Values are stored in mpi_strnbkt.

**NICKNAME**

Using the nickname string enables you map names with alternate names, thus increasing the accuracy of comparisons. Example mappings may include: Abigail to Abbey or Gail, William to Bill or Willie, or Thomas to Tom, Tommy, or Tommie. Values are stored in mpi_strequi.

**PHONE**

As with Social Security number, dummy values may be entered in the phone field. Specifying such values prevents them from being used in comparisons. Values are stored in mpi_stranon.

**PXNM-DGR**

This string enables you to define additional standardization for values that might appear in a member name field; for example CPA, DDS, or ESQ. Values are stored in mpi_strword. 
**PXNM-FNJ**

This string enables you to define additional standardization for values that might appear in a member's first name; for example “Lo” or “La.” Values are stored in mpi_strword.

**PXNM-LNJ**

This string enables you to define additional standardization for values that might appear in a member's last name; for example “Ben” or “Von.” Values are stored in mpi_strword.

**PXNM-MNJ**

This string enables you to define additional standardization for values that might appear in a member's middle name; for example “Jo” or “La.” Values are stored in mpi_strword.

**PXNM-PFX**

This string enables you to define additional standardization for values that might appear as a prefix in a member name; for example “Dr” or “Mrs.” Values are stored in mpi_strword.

**PXNM-SFX**

This string enables you to define additional standardization for values that might appear as a suffix in a member name; for example “Jr” or “II.” Values are stored in mpi_strword.

**QXNM-NAME**

You have the option to specify RealNames in mpi_strword using a strcode of QXNM-NAME. The words specified in this table are not compared for phonetic or edit-distance matches. All real names need to exist in mpi_strword with a strcode of QXNM-NAME. See “Enabling RealNames with QXNM” on page 137.

**RXNMRABICRULES**

This string provides an example set of ENCODER rules for processing Arabic names with RXNM. See “Custom Phonetics and Rule Sets” on page 281 for more information.

**SSA**

Again, often when entering a member record the Social Security number may not be available and dummy values are entered (e.g., 999999999). By specifying such values in this string, the algorithm can identify which values to ignore during comparison. Values are stored in mpi_stranon.

**WGT**

For implementations using member weight comparisons, this string is used to generate a standardized member weight value (range of weights) for bucketing and comparison. Bucketing and comparing on exact weight values can be difficult as the weight entered for a person may be just an approximation. Using the WGT...
String, you can define weight ranges. For example, you might specify that all weight values of 100 pounds to 109 pounds should have a bucket value of 100 pounds, and 110 to 119 should have a bucket value of 110. Values are stored in mpi_strnbkt.

ZIPCODE

Dummy values may be entered when this information is not available during member record creation. Specifying dummy Zip Code values prevents them from being used in comparisons. Values are stored in mpi_stranon.

Enabling RealNames with QXNM

About this task

All real names need to exist in mpi_strword with a strcode of QXNM-NAME.

To enable RealNames with QXNM:

Procedure

1. Copy the real names string file from the Demo Person Project
   \workbench_install\plugins\com.initiatesystems.workbench_###VB###ReleaseNumber##
   \VE###.0\projectTemplates\Initiate_Demo_Person\strings\word\qxnm-name.txt
   to the strings\word folder in your active IBM Initiate Workbench project:
   workspace\projectname\strings\word
2. On the Strings view in the Hub Configuration editor, create a new string code
   named QXNM-NAME using the WORD Type.
3. You will then be able to choose qxnm-name.txt from the String value file
   pull-down list. If it does not appear, right-click the project in the Navigator
   view and select Refresh.
4. Deploy your configuration to populate the Hub with the real name values.

String codes

About this task

In the Strings view, select the desired string from the String code pull-down list.
The string details appear in the Properties view. The string values associated with
the string code are displayed below the String value file field.

Adding a string

Procedure

1. In the Strings view, click Add. A new string is added with default property values.
2. In the Properties view, set the string’s Type. This selection determines the
   database table in which the values will be stored. Types include:
   • ANON – anonymous values. Use this when defining values you want the
     algorithm to ignore during comparison.
   • EQUI – equivalent values. Use this to standardize variations in incoming
     data by telling the software to map x value to y. Used primarily for specialty
     codes or nicknames.
   • FREQ – frequency values. Use this to define a limitation on the number of
     times the same string value will be bucketed.
   • WORD – word values. Use this to define standardized values for words used
     in addresses, business names, and member names (e.g., “Von” in last names).
• NBKT – numeric values. Use this to define standardized values for numeric bucketing.
• SBKT – string values. Use this when defining string values that are
  standardized to a common value for the purpose of bucketing.

3. Type a Code, Name, and Description of the string. This information is stored
  in MPI_strhead.

4. The String Value File field is read-only. Select a different file using the String
  value file pull-down list. To add a string value file, follow the instructions
  under “Adding string code values.”

5. Save the project.

Editing a string

Most of the properties for a String code can be edited by following this process.

Procedure
1. In the Strings view, highlight the desired string.
2. In the Properties view, make any necessary changes in the String Info area.
3. The String Value File field is read-only. Select a different file using the String
   value file pull-down list. To add or edit string value files, follow the
   instructions under “Adding string code values” below, and “Editing string code
   values” on page 139.
4. Save the project.

Deleting a string

About this task

When you delete a string, the string value file is also deleted.

Procedure
1. In the Strings view, select the unwanted string code from the pull-down list,
   and then click Remove.
2. Save the project.

Adding string code values

About this task

The string code value files that are available depend on the string code's Type. To
create a new string code value file:

Procedure
1. In the Strings view, click New.... The New String Value File dialog opens.
2. Enter the name of the text file you wish to use to store the string code values.
   This file must end with a .txt filename extension.
3. Click OK. The String value file is created and now appears in the pull-down
   list of string value files for the selected String code.
4. Click Edit.... The string value file opens in the editor and you can begin to add
   values. Use the vertical bar (|) character to separate fields. The # character
   denotes a line used to label the fields in the file.

Note: Refer to “String file fields to be defined” on page 139 for the fields to
specify for each string type.
The fields in the file will differ depending on the Type of the string code selected.

5. If you are creating a string code value file for anonymous addresses, the address values must be standardized. Refer to “Standardizing anonymous address strings” on page 141.

6. Save the file and close its editor tab when you are finished adding values. The new values now appear in the Strings view.

**Editing string code values**
The String code value file can be edited by following this process. Use this process to add, change or delete elements to an existing string code value file.

**Procedure**
1. From the pull-down list of String value files, select the file to be edited.
2. Click Edit... The string value file opens in the editor with the existing values displayed. Use the vertical bar (|) character to separate fields.
3. If you are editing a string code value file for anonymous addresses, the address values must be standardized. Refer to “Standardizing anonymous address strings” on page 141.
4. Save the file and close its editor tab when you are finished editing values. The new values now appear in the Strings view.

**Removing duplicate data from string code values file:**

**Procedure**
1. From the pull-down list of String value files, select the file to be cleaned.
2. Click Dedup. The utility runs in the background to remove duplicate entries within the string code values file.
3. Click OK to close the Dedup String Value File message box. You can verify that the duplicates have been removed by editing the string code values file.

**String file fields to be defined**

This section contains a list of the fields to be defined for a string value file. Refer to this list when adding or editing a string value file.

*Table 46. String file fields to be defined*

<table>
<thead>
<tr>
<th>If you are adding type...</th>
<th>...then define the following</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANON</td>
<td>Value: The value to be ignored.</td>
</tr>
</tbody>
</table>
| CMAP                      | CMAP string types are used for translations at the character level. They appear in this format:  
0001;0020 or  
0001..001F;0020  
Index: A simple index.  
Value: Value (or range of values) to be substituted; substitute value |
<table>
<thead>
<tr>
<th>If you are adding type...</th>
<th>...then define the following</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CONFIG</strong></td>
<td><strong>CONFIG</strong> string types are used for custom phonetic functions. Refer to “Custom Phonetics and Rule Sets” on page 281 for information on custom phonetic functions and the syntax to follow.</td>
</tr>
<tr>
<td></td>
<td><strong>Index value</strong>: A simple index.</td>
</tr>
<tr>
<td></td>
<td><strong>Value</strong>: The rule set that defines the encoding for the string. Refer to XXX for information on the custom phonetic function rules and rule sets.</td>
</tr>
<tr>
<td><strong>EDIT</strong></td>
<td><strong>EDIT</strong> string types are used to map one value to another.</td>
</tr>
<tr>
<td></td>
<td><strong>Input value</strong>: The value to be substituted.</td>
</tr>
<tr>
<td></td>
<td><strong>Output value</strong>: The substitute value.</td>
</tr>
<tr>
<td><strong>EQUI</strong></td>
<td><strong>Value</strong>: The incoming value.</td>
</tr>
<tr>
<td></td>
<td><strong>Alternate Value</strong>: Alternate equivalent value.</td>
</tr>
<tr>
<td><strong>FREQ</strong></td>
<td><strong>Value</strong>: The incoming value.</td>
</tr>
<tr>
<td></td>
<td><strong>Frequency</strong>: The number of times the string can be bucketed.</td>
</tr>
<tr>
<td><strong>NBKT</strong></td>
<td><strong>Numeric Value</strong>: The incoming value.</td>
</tr>
<tr>
<td></td>
<td><strong>Bucket Value</strong>: This is how the value should appear after standardization.</td>
</tr>
<tr>
<td><strong>SBKT</strong></td>
<td><strong>Value</strong>: The incoming value.</td>
</tr>
<tr>
<td></td>
<td><strong>Bucket Value</strong>: This is how the value should appear after standardization.</td>
</tr>
<tr>
<td><strong>SET</strong></td>
<td>Used in the weight adjustment algorithm when unit types such as APT, UNIT, #, and so forth are present. The value is implementation-defined.</td>
</tr>
<tr>
<td></td>
<td><strong>Value</strong>: A string contained in the set.</td>
</tr>
</tbody>
</table>
Table 46. String file fields to be defined (continued)

<table>
<thead>
<tr>
<th>If you are adding type...</th>
<th>...then define the following</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD</td>
<td>Value: The incoming value.</td>
</tr>
<tr>
<td></td>
<td>Standard Value: This is how the value should appear after standardization and the value that is used for comparison and bucketing.</td>
</tr>
<tr>
<td></td>
<td>Word Type: The following lists valid word types and the string types in which they are normally used.</td>
</tr>
<tr>
<td></td>
<td>ST: street; normally used in ADDR-TOK</td>
</tr>
<tr>
<td></td>
<td>UT: unit such as apartment or suite; normally used in ADDR-TOK</td>
</tr>
<tr>
<td></td>
<td>MT: mail type, such as P.O. Box; normally used in ADDR-TOK</td>
</tr>
<tr>
<td></td>
<td>BN: block number; normally used in ADDR-TOK</td>
</tr>
<tr>
<td></td>
<td>ANT: alpha-numeric token; user in ADDR-TOK and PXNM</td>
</tr>
<tr>
<td></td>
<td>DT: direction; normally used in ADDR-TOK</td>
</tr>
<tr>
<td></td>
<td>BLT: business legal type; normally used in BXMN-TOK</td>
</tr>
<tr>
<td></td>
<td>PCT: person credential type; normally used in BXMN-TOK</td>
</tr>
</tbody>
</table>

Standardizing anonymous address strings  
About this task  
The anonymous addresses you provide in the addr2-manon.txt file must be in standardized format. To get a standardized address, you can use the mpimshow utility.

Procedure  
1. Run an Analytics report to retrieve the MEMRECNO of the member whose address you wish to designate as anonymous. Refer to Chapter 7, “Analytics,” on page 187 for more information.
2. On the computer hosting the Hub, open a command line window and navigate to the hub\install\path\bin directory.
3. Run the mpimshow utility, specifying the MEMRECNO retrieved in step 1, and direct the output to a file in the hub\instance\path\inst\mpinet_hub_instance_name\work\project_name\work\directory. For example
   mpimshow -html 1 >
   c:\myhub\inst\mpinet_myhub\work\myproject\myworkdir\mem1.html
   or
   mpimshow -xml 1 >
   c:\myhub\inst\mpinet_myhub\work\myproject\myworkdir\mem1.xml
   For details on the syntax, consult the IBM Initiate Master Data Service Engine Installation Guide. The output is saved to a file in the specified location.
4. In IBM Initiate Workbench, retrieve the output file by using the Get File job. Refer to “Get File” on page 87.
5. Copy the address value (the cmpval string) from the mpimshow output file, and paste it into the addr2-manon.txt file.
Anonymous Value Utility

The Anonymous Value Utility is used in the weight generation process (refer to Chapter 5, “Algorithm editor,” on page 155) to simplify the selection of anonymous values from a list of candidate attribute values. The utility is launched from the Strings tab in the Configuration editor by clicking Run Anonymous Value Utility. This button is enabled when the selected String Code’s Type property is set to ANON.

The Anonymous Value Utility uses the mpi_strfreq.unl and mpi_strhead.unl files which are created by running the mpxfreq job (see “Generate Frequency Stats (mpxfreq)” on page 27), with the “Generate frequency tables for the Anonymous Value Utility” option. When the job is done, the mpi_strfreq.unl and mpi_strhead.unl files can be retrieved from the Hub and copied into the project by running the “Get job results” action from the Jobs view. The mpi_strfreq.unl file contains attribute values and their counts, while mpi_strhead.unl contains the total number of records in the data set.

Note: Because mpxfreq does not support the counting of multi-token attributes, such as addresses, it cannot provide the Anonymous Value Utility the information it needs to designate addresses as anonymous. This means anonymous addresses must be manually added to the ANON string value file associated with ADDR2-MANON. See “Adding string code values” on page 138 and “Standardizing anonymous address strings” on page 141 for more information.

The utility displays the anonymous string codes, associated standardization roles, and attribute values (i.e., candidate lists), grouped by their anonymous string code. The user selects attribute values from the candidate lists which are determined to be anonymous values. Each candidate list can be filtered using various methods, such as by minimum frequency, to narrow down the choices in the list.

In addition to selections in the candidate lists, default anonymous values can be specified for inclusion when the selections are saved.

Default value files and filter files

The default value files (*.def) and filter files (*.fil) will always be treated as UTF8 by the utility. This encoding offers the best flexibility, allowing the lists to contain both Unicode and non-Unicode characters. However, in the case of the default value files, a default anonymous value consisting of Unicode characters cannot be added to an mpi_stranon.sql file which has the LATIN1 encoding format without causing an error.

The Default Values Editor automatically reads and writes the default value files with the UTF8 encoding. However, if these files, or the filter files, are edited with an external editor and contain Unicode characters, they must be saved with the UTF8 encoding.

Loading frequency data

In order to use the Anonymous Value Utility, you need to first run the Generate Frequency Stats (mpxfreq) job and retrieve the results.

Procedure

1. Run the mpxfreq utility using the New Job Set wizard with the “Generate frequency tables for the Anonymous Value Utility” option enabled. See Chapter 3, “Jobs and job sets,” on page 21 for details.
2. When the mpxfreq job has successfully finished, right-click the job and select Get job results. A new folder (frq) is created in the IBM Initiate Workbench project and the frequency data is copied into it.

**Launching the utility**

To use the Anonymous Value Utility, you run it from the Strings view.

**Procedure**
1. In IBM Initiate Workbench, click the Strings tab to open the Strings view.
2. Select a string code whose type is ANON.
3. Click Run anonymous value utility. The Anonymous Value Utility dialog opens.

**Results**

Note that if no anonymous string codes are referenced by the frequency input data, the Anonymous string code list will be empty and no candidates will be displayed.

**Using the utility**

Anonymous value string codes are selected in the Derivation code area of the screen.

**Procedure**
1. On the Anonymous Value Utility dialog’s Select resources page, click Browse beside the Frequency folder field to select the frq folder in the IBM Initiate Workbench project. Click Next.
2. Select the desired Anonymous string code from the list. The default value is the string code that was selected when you started the utility.
   When you select an anonymous string code:
   - The Standardization roles field displays a comma-delimited list of standardization roles that reference the string code (e.g., LGLNAME).
   - The associated tab in the Anonymous value candidate list area is populated with the attribute values from the frequency input data. The attribute counts and calculated frequency percentages are provided.
3. Optional: If you wish to export the attribute frequency data to a *.csv file, click Export. The Data Export dialog opens.
   a. Indicate the directory in which to save the file. The default is the local project directory, but any accessible path is permitted by clicking Browse.
   b. Indicate the desired field separator (the comma is the default separator) and whether you wish to include a column header row or comments and associated parameters.
   c. Click Finish to save the file.
4. Indicate whether to Include default values. Default anonymous values are lists of pre-defined values that have been determined to always be anonymous. For example, “Baby Boy” is often defined as an anonymous name or “999999999” for Social Security numbers. An example set of default values can be found in workspace\project_name\anonutil\defaults (files with a .def file extension).
5. If default values are to be included, select the default value code from the pull-down list.
   a. If you need to edit the list, click Edit... The Edit Default Anonymous Values dialog opens.
b. Type the new value in the list. Specify each value on a separate line. If needed, create a new Default value code by clicking New.... Type the new code in the provided field and click OK. The name can only contain alpha-numeric characters, hyphens and dashes, and it must be unique.

c. Click Close. The editor detects if any changes have been made to the existing list of default values or if new default codes and value lists have been created.

d. You are prompted to save or ignore the changes. If you select Yes, the value lists are persisted to the file system using the default value code as the filename with a .def extension.

Note: The Anonymous Value Utility does not enable you to delete a default value code. However, since the codes are derived from the file names of the .def files, you can delete (or rename) a default value code and associated list by deleting (or renaming) the file in the \defaults directory.

6. Select the desired filter by clicking the associated radio button. Filtering enables you to narrow the selections in a candidate list. The default filter setting is Minimum frequency with a minimum frequency of 0.1%.

No filter
Displays all values in the candidate list.

Minimum frequency
This filter is used on generic data. Only values with a frequency greater than or equal to the minimum frequency number are displayed. Minimum frequency ranges are 0.1 to 5.0 percent. You can use the UP and DOWN arrows on your keyboard to increase or decrease by 0.1%, while using PAGE UP and PAGE DOWN to increase/decrease by 1.0%.

Filter list plus minimum frequency
Select an option from the pull-down list and then specify a minimum frequency percentage (from 0.1 to 5.0). This option enables you to specify a list of values that should be eliminated from the candidate list. For example, there are many names that occur with high frequency, such as John, Mary, Smith, and Jones. Including these names in the list can make it more difficult to find true anonymous values. You can filter by first name and set the minimum frequency percentage to a high number to filter out John and Mary. You can use the UP and DOWN arrows on your keyboard to increase or decrease by 0.1%, while using PAGE UP and PAGE DOWN to increase/decrease by 1.0%.

Two example filter lists (one for common English first names and one for common last names) can be found in workspace\project_name\anontool\filters. Filter list files have a .fil extension.

Neighbor frequency
This filter is used mainly for dates. The list of values is sorted and displays a value whose frequency is greater than or equal to the average frequency of the values before and after multiplication by a specified number. The multiplier value range is 0 to 20, with a default of 10. You can use the UP and DOWN arrows on your keyboard to increase or decrease by 1, while using PAGE UP and PAGE DOWN to increase/decrease by 5.

7. Select an attribute value from the Anonymous value candidate lists by doing one of the following:
   • Double-click a single row.
• Highlight one or more rows in the list and click the checkbox in the Anonymous column.
• Highlight one or more rows and right-click. Select Make anonymous or Make non-anonymous from the menu.

The Anonymous value candidate lists display attribute values, counts and calculated frequencies. Each candidate list is grouped by the anonymous string code. When a selection is made in the Anonymous string code field at the top, the grid displays the associated candidate list.

When a selection is made in the candidate list, the attribute value is designated as an anonymous value when the data is saved.

The sort order of the candidate lists can be changed by clicking one of the column headings (e.g., Attribute Value, Count, and so forth).

8. Click Finish. The Save Anonymous Values dialog opens.
9. You can select whether you want to Merge with existing anonymous values. This option merges your changes with the existing values saved locally. Duplicate values are eliminated during the merge process. If you select Overwrite existing anonymous values, any previous changes saved locally are overwritten.
10. Click OK.

Groups

Security measures can be implemented through IBM Initiate Workbench: by setting up user accounts and defining permission levels on a user group basis. See Chapter 9, “User management,” on page 207.

Through the LDAP perspective, you can create users and assign them to groups (see Chapter 9, “User management,” on page 207). Through the Groups component you can establish authorization and permission levels for your LDAP groups.

To access the Groups component, click the Groups tab. In the Groups view, you can access options by clicking the Composite views, Interactions, Attributes/Sources, Operations and Relationship attributes sub-tabs.

Note: Until a new user is added to the Administrators group (cn=Administrators,ou=System,ou=Groups, dc=Initiatesystems,dc=com), you must use the system user name and password to execute the group synchronization tool. Once a IBM Initiate Workbench user is added to the Administrators group, he can use his own login credentials to continue with Hub configuration.

To assign users to groups, add users and assign them to groups as described in Chapter 9, “User management,” on page 207 and assign group permissions on the Groups tab.

Running the Synchronize Group Definitions wizard

About this task

This wizard sends a request to the Hub for the current list of user groups. The Hub then queries the LDAP server for the list and returns it to IBM Initiate Workbench to display on the Groups tab. Since LDAP users and groups may be updated at any time, it is a good practice to periodically refresh the list of available groups before beginning the task of setting permissions.
Procedure
1. Click the Groups tab to open the Groups screen.
2. Click Refresh Groups. The Synchronize Group Definitions wizard opens.
3. Select the Hub from which you wish to retrieve group definitions. If you need to change the connection settings for the Hub, or add a new Hub, click Edit and refer to “Hub connections” on page 11.
4. Click Next. You are asked to log in. Here you must supply an LDAP user name and password which belongs to the (cn=Administrators,ou=System,ou=Groups,dc=Initiatesystems,dc=com) group.
   • If the groups listed in the Groups pull-down list on the Groups tab matches the LDAP server to which the Hub is connected, no action is taken. Skip to step 6.
   • If the group list has changed on the LDAP server, a list of the groups that are to be added or deleted from the list in IBM Initiate Workbench is displayed. For groups that were added to the LDAP server, you can click the right column, “Copy Permissions from Group,” to select an existing group from which to copy all permissions. Leave this blank if you do not wish to copy permissions.

Note: Groups that were renamed on the LDAP server are treated as an add and delete operation for purposes of updating the IBM Initiate Workbench groups list. If you want to preserve any permissions of an IBM Initiate Workbench group that is to be deleted you will want to copy its permissions to a group to be added in the Copy Permissions from Group column.
5. Click Next. A list of groups to be added or deleted is shown.
6. Click Finish.

Group permissions
Establishing permissions for user groups for interactions and attribute types (segments) enables you to further implement security and control by assigning users to the group(s) that best fit their individual job requirements. For example, you may have users who need to look at member records and understand what records are involved in tasks, but not actually work the tasks. In that case, you could create a group with permissions for member searches, member retrieves, and task searches. A group of users responsible for working tasks would need additional permissions that enabled them to edit records, tasks, and attributes.

Users can be affiliated with more than one group. If a user belongs to multiple groups, their permissions are a union of the groups with the highest permission level being used.

Setting permissions for composite views
Setting group permissions for composite views defines which users can access the composite views.

Procedure
1. Click the Groups tab to open the Groups screen.
2. Select the Group for which you wish to set permissions. To retrieve the most up-to-date list of groups, refer to “Running the Synchronize Group Definitions wizard” on page 145.
3. Click the Composite Views tab.
4. Check or uncheck the box beside the desired composite view to grant or remove access. To enable or disable multiple interactions, shift-click or ctrl-click to highlight the desired interactions, then click a selected item's checkbox.

Results

The composite views to enable for an application are often defined in the application's properties file. For example, IBM Initiate Inspector properties may include lines similar to the following:

```
inspector.entity.composite.view=EMCA
inspector.member.composite.view=MMCA
```

In this case, IBM Initiate Inspector users probably need permissions for both composite views. WebReports do not use composite views.

Setting permissions for interactions

Interactions are the functions one is allowed to perform, such as compare members, update attributes, search for members, add members, merge members, or update tasks. Setting permissions for interactions enables you to restrict or allow these functions for groups of users.

Procedure

1. Click the Groups tab to open the Groups.
2. Select the Group for which you wish to set permissions. To retrieve the most up-to-date list of groups, refer to "Running the Synchronize Group Definitions wizard" on page 145.
3. Click the Interactions tab.
4. Check or uncheck the box beside the desired interaction to grant or remove access. To enable or disable multiple interactions, shift-click or ctrl-click to highlight the desired interactions, then click a selected item's checkbox.

Related reference:

"Interaction guidelines"

Interaction guidelines

Refer to the following table for guidelines on selecting interactions.

Refer to the IBM Initiate Master Data Service SDK Reference for Java and Web Services for a description of all interactions.

**Table 47. Interaction guidelines**

<table>
<thead>
<tr>
<th>Users of...</th>
<th>Usually need the following interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise Viewer</td>
<td>Get Application Data (GETAPPINFO)</td>
</tr>
<tr>
<td></td>
<td>Get User Data (USRGETINFO)</td>
</tr>
<tr>
<td></td>
<td>Get Dictionary Data (DICGET)</td>
</tr>
<tr>
<td></td>
<td>Get Member (MEMGET)</td>
</tr>
<tr>
<td></td>
<td>Update Member (MEMPUT)</td>
</tr>
<tr>
<td></td>
<td>Manage Member Notes (MEMPOSTIT)</td>
</tr>
<tr>
<td></td>
<td>Search Members (MEMSEARCH)</td>
</tr>
<tr>
<td></td>
<td>Get Task (TSKGET)</td>
</tr>
</tbody>
</table>
Table 47. Interaction guidelines (continued)

<table>
<thead>
<tr>
<th>Users of...</th>
<th>Usually need the following interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Initiate</td>
<td>Get Application Data (GETAPPINFO)</td>
</tr>
<tr>
<td>Inspector</td>
<td>Get User Data (USRGETINFO)</td>
</tr>
<tr>
<td></td>
<td>Get Dictionary Data (DICGET)</td>
</tr>
<tr>
<td></td>
<td>Get Member (MEMGET)</td>
</tr>
<tr>
<td></td>
<td>Update Member (MEMPUT)</td>
</tr>
<tr>
<td></td>
<td>Manage Member Notes (MEMPOSTIT)</td>
</tr>
<tr>
<td></td>
<td>Search Members (MEMSEARCH)</td>
</tr>
<tr>
<td></td>
<td>Get Task (TSKGET)</td>
</tr>
<tr>
<td></td>
<td>Special (RELGET)</td>
</tr>
<tr>
<td></td>
<td>Special (RELSEARCHCOUNT)</td>
</tr>
<tr>
<td></td>
<td>Special (RELPUT)</td>
</tr>
<tr>
<td></td>
<td>Special (RELDELETE)</td>
</tr>
<tr>
<td></td>
<td>Relationship Task Search (RELSKSEARCH)</td>
</tr>
<tr>
<td></td>
<td>Relationship Task Resolve (RELSKRESOLVE)</td>
</tr>
<tr>
<td></td>
<td>Relationship Task Delete (RELSKDELETE)</td>
</tr>
<tr>
<td></td>
<td>Relationship Task Get (RELSKGET)</td>
</tr>
<tr>
<td></td>
<td>Relationship Search (RELSKSEARCH)</td>
</tr>
<tr>
<td></td>
<td>Relationship Path to Root (RELPATHTOROOT)</td>
</tr>
<tr>
<td>Web Reports</td>
<td>Get Application Data (GETAPPINFO)</td>
</tr>
<tr>
<td></td>
<td>Get User Data (USRGETINFO)</td>
</tr>
<tr>
<td></td>
<td>Get Dictionary Data (DICGET)</td>
</tr>
<tr>
<td></td>
<td>Get Member (MEMGET)</td>
</tr>
<tr>
<td></td>
<td>Search Members (MEMSEARCH)</td>
</tr>
<tr>
<td></td>
<td>Get Task (TSKGET)</td>
</tr>
</tbody>
</table>

Setting permissions for attribute types
Attribute type (segment) permissions determine whether a user can view information (Read Only) or view and update information (Read-Write). For example, Read-Write permissions enable IBM Initiate Inspector users to perform edits and updates to records, such as edit attribute status, surviving Enterprise ID and Source ID numbers, and add members to the hub database.

About this task

A Read Only user can search and view member records, but cannot work tasks.

Procedure

1. Click the Groups tab to open the Groups screen.
2. Select the Group for which you wish to set permissions. To retrieve the most up-to-date list of groups, refer to “Running the Synchronize Group Definitions wizard” on page 145.
3. Click the Attributes/Sources tab. The information is displayed in a tree form hierarchy, with the member type at the top and either Sources or Attributes as the second level. The Tree Organization option on the right side lets you specify whether to display attributes or sources as the second level.
   • When “sources” is the second level, each source is listed for the member type; expand a source to view the attributes configured for that source.
   • When “attributes” is the second level, each attribute is listed for the member type; expand an attribute to view the sources configured for that attribute.
4. Set the **Source access privileges** for the second-level item as needed to control the default access privileges for all sources or attributes under that item. You can then individually set the access privileges for specific items as desired.

Changing the Tree Organization option enables you to verify your changes and view the access privileges that were granted for a specific attribute or source.

**Setting permissions for operations**
Operations permissions determine whether a user can execute certain operations that are specific to an application.

**Procedure**
1. Click the Groups tab to open the Groups screen.
2. Select the Group for which you wish to set permissions. To retrieve the most up-to-date list of groups, refer to "Running the Synchronize Group Definitions wizard" on page 145.
3. Click the **Operations** tab. Refer to "Operations Guidelines" for a list of operations available for the various applications.
4. Set or clear the checkbox for each Application/Operation combination to enable or disable access to the application and its specific operation for the group.

**Operations Guidelines**
Refer to the following information for guidelines on selecting operations for user groups.

**Workbench**
- **jmxAdmin**
  enables group members to view and execute all JMX MBeans, including Initiate-specific operations as well as any other Java-specific operation.
- **jmxExecute**
  enables group members to view and execute all Initiate-specific JMX MBeans, including job execution.
- **jmxMonitor**
  enables group members to view and execute certain Initiate-specific JMX MBeans, excluding job execution.

**Inspector**
- **inspectorAdmin**
  enables group members to execute certain administrative functions specific to the IBM Initiate Inspector application.

**Data Trust**
- **crAdmin**
  enables group members to administer change requests for all users and groups in the system.
- **createEntity**
  Granting group privileges to the "createEntity" operation will enable the "Create Entity" link in the Data Trust user interface, enabling group members to insert a record into the system.
relAdmin
Granting group privileges to the “relAdmin” operation enables the “Upload”
link in the Data Trust user interface, enabling group members to import
relationships into the system.

Identity Hub Reports

auditorDuplicationSummaryStatistics
enables group members to access the Duplication Summary Statistics report.

auditorEventActivity
enables group members to access the Event Activity Report.

auditorEventDetail
enables group members to access the Event Detail Report.

auditorEventSummary
enables group members to access the Event Summary Report.

auditorLinkageManagementOverview
enables group members to access the Linkage Management Overview report.

auditorNotesDetail
enables group members to access the Notes Detail report.

auditorPendingResolutionDetail
enables group members to access the Premerge Report.

auditorRecordResolutionDetail
enables group members to access the Record Resolution Detail report.

auditorTaskCreationDetail
enables group members to access the Task Creation Detail report.

auditorTaskManagementOverview
enables group members to access the Task Management Overview report.

hubmgrInitiateAdditions
enables group members to access the Initiate Additions report.

hubmgrUserActivitySummary
enables group members to access the User Activity Summary report.

operationalAssignedTaskCountByOwner
enables group members to access the Task Assignment By Owner and Type
report.

operationalOutstandingTaskCountBySource
enables group members to access the Outstanding Tasks By Source and Type
report

operationalTaskCountByTagType
enables group members to access the Task Count By Type report.

Setting permissions for relationship attributes
You can set read-write, read-only or no access permissions for users to view
relationship attributes.

Procedure
1. Click the Groups tab to open the Groups screen.
2. Select the Group for which you wish to set permissions. To retrieve the most
   up-to-date list of groups, refer to “Running the Synchronize Group Definitions
   wizard” on page 145.
3. Click the **Relationship Attributes** tab. The information is displayed in a tree form hierarchy, with the relationship type at the top and attributes as the second level.

4. Set the **Access Privileges** for the Relationship Type item as needed to control the default access privileges for all of its attributes. You can then individually set the access privileges for specific attributes as desired.

**Assigning or removing a user from a group**

To remove a user from a group, use the LDAP perspective to set the user's group(s).

**Related concepts:**

Chapter 9, “User management,” on page 207

### Auditing view

The Auditing tab enables you to establish audit logging for Initiate interactions and the users performing these interactions. Logging is set individually for each interaction and can be set at one of three levels:

- **None** turns the logging off.
- **Activity** logs the user or system name performing the interaction along with the date and time.
- **Member** is the most extensive level of logging. This means that the user/system name, date, time, member record affected, and value changes are logged.

For example, assume that the MemPut interaction is set to “Member.” IBM Initiate Inspector user “johnbell” retrieves a record for Patty Countryman. During the review of the record, johnbell changes the home phone number attribute and saves the record. A record of this transaction is written in the Hub database MPI_audxmem table.

You can view audit information in directly from the database via SQL. The following is sample SQL that can be used to extract the information:

```sql
select u.usrlogin as "User ID", a.audctime as "Activity Time", 
    s.srccode as "Source", h.memidnum "ID", i.ixnlabel as "Activity"
from mpi_usrhead u, mpi_audhead a, mpi_srchead s, 
    mpi_memhead h, mpi_audxmem x, mpi_ixnhead i
where u.usrrecno = a.usrrecno 
and x.audrecno = a.audrecno 
and x.memrecno = h.memrecno 
and s.srcrecno = h.srcrecno 
and i.ixnrecno = a.ixnrecno 
```

To limit the results to a single user, add the following to the end of the above statement:

```sql
and a.usrid = 'rwuser'
```

The output from the above query (requesting all users) will be similar to:

```
rwuser 2005-07-29 11:33:15.000 RMC 870504 Search Members
rwuser 2005-07-29 11:33:15.000 RMC 558090 Search Members
rwuser 2005-07-29 11:33:15.000 PHYS 76690 Search Members
rwuser 2005-07-29 11:33:15.000 PHYS 899000 Search Members
billp 2005-07-29 11:33:15.000 ARH 1003702 Search Members
rwuser 2005-07-29 11:33:43.000 RMC 279776 Search Members
rwuser 2005-07-29 11:33:43.000 RMC 788888 Search Members
billp 2005-07-29 11:33:43.000 FGH 260765 Search Members
```
Setting user audit logging

To record a history of user interactions, you set audit logging levels.

Procedure
1. Click the Auditing tab.
2. Click on the existing Access privileges for an interaction.
3. Select the desired access level from the pull-down list.
4. Save the project.

Results

The hub configuration must be deployed to the Master Data Engine to recognize the changes to audit logging.
Chapter 5. Algorithm editor

The algorithm is the brain of the IBM Initiate Master Data Service software, therefore, the proper data elements must be in place before you can fully develop the algorithm. After your first pass at the implementation, you may make some tweaks to the algorithm. After making those changes, you will need to derive data, generate weights and/or perform a Bulk Cross Match again.

You can view differences between algorithms in a IBM Initiate Workbench project and those on a Hub by using the Configuration Comparison feature. Refer to "Configuration comparison view” on page 7 for more information.

Sticky selection

The Set palette tool selection sticky option enables you to place multiple copies of a palette object on the editor without having to re-click the object on the palette. Palette objects affected by this setting are: Comparison Role, Query Role and Bucketing Group. The Connection object is always sticky.

**Note:** If the algorithm editor is open when the Set palette tool selection sticky option is changed, you will need to close the editor and reopen it for the setting to take effect.

To enable "sticky" object selection in the palette, select **Window > Preferences > Initiate > Algorithm Editor**.

Swimlane width

The column or "swimlane" width in the Algorithm Editor can be adjusted as desired to accommodate your screen resolution.

To adjust "swimlane" sizes in the editor, select **Window > Preferences > Initiate > Algorithm Editor**. Moving the **Swimlane width** slider to the left narrows the columns; moving it to the right widens them. Click **OK** or **Apply** to apply the change.

Classic vs. composite algorithm function names

Some of the classic function names shown in the Algorithm Editor may seem somewhat obscure to new users. The **Use classic functions** option determines whether the classic names or composite names are used. For example, the comparison function DR1D1A would be listed in the Algorithm Editor as “Edit distance” when Use classic functions is disabled. As described in the **Appendix A, Algorithm function descriptions,” on page 247** there are several edit distance functions, differentiated by the number of dimensions and whether they are ‘quick,’ ‘full’ or ‘simple.’ When you use Composite mode, these discriminators are specified using the Kind and Type properties.

To enable or disable the **Use classic functions** option, select **Window > Preferences > Initiate > Algorithm Editor**. The default is disabled, which displays the composite (plain English) names.
Note that the functions described in the Appendix A, “Algorithm function descriptions,” on page 247 are listed by their classic names. A set of mapping tables (Table 67 on page 247 Table 68 on page 248 Table 69 on page 249) will help you identify the functions called by their composite names and types.

Algorithm and Member Type associations

The algorithms that appear in the Navigator are based on the member types currently defined in the Hub configuration. If you want to create a new algorithm, you must associate it with a member type in the Hub configuration. Refer to Chapter 4, “Configuration editor,” on page 89 for details. When an algorithm is initially created, it is empty. You can add algorithm components and connections from the Palette in the Algorithm editor to construct the algorithm.

Algorithms are typically named according to the member type names with which they are associated, followed by an .ALG extension. Algorithms which are imported from an existing Hub configuration will be initially named based on their associated member type. For example, an algorithm for the Person member type will be named "PERSON.alg" when it is imported. The algorithm can be renamed via the Hub configuration editor in the Algorithms tab, under the Member Types tab, and the name does not have to match the member type. In fact, multiple algorithms can be associated with a particular member type, although only one can be set as the "active" algorithm at any given time.

Loading an algorithm

About this task

Algorithms are identified in the Navigator by a .alg extension. Double-clicking an algorithm in the Navigator opens it in the editor. If your project does not yet contain an algorithm, follow the steps below.

Note: If you are configuring an algorithm for both searching and matching/linking, you may find it helpful to use the Tuning Search cheat sheet. Refer to Appendix E, “Creating cheat sheets for Initiate tools,” on page 349 for more information.

Procedure

1. Open an Initiate project containing the Hub configuration. Refer to Chapter 2, “Managing projects and Hub connections,” on page 11 for detailed instructions on creating projects and importing Hub configurations.
2. In the Navigator pane, double-click the desired algorithm to open the algorithm in the editor view.

Saving an algorithm

About this task

After making desired changes to an algorithm, it must be saved to the local file system. In order for an algorithm to be uploaded to a Master Data Engine for use in a test or production environment, its IBM Initiate Workbench project is deployed to that server. Refer to Chapter 2, “Managing projects and Hub connections,” on page 11 for detailed information on deploying projects.

To save an algorithm, use one of the following actions:
• Click the **Save** icon on the IBM Initiate Workbench toolbar.
• Press Ctrl+S on the keyboard.
• Select **File > Save** from the main menu.

Algorithms are stored locally by IBM Initiate Workbench in the workspace directory in a child directory under the project. The workspace directory is created when IBM Initiate Workbench is started. Typically the algorithms are saved as:

..\workspace\project_name\algorithm_name.alg

For example, an algorithm based on the Guest member type that is in project Alpha would typically be saved to the following path:

C:\Documents and Settings\current_user\workspace\Alpha\Guest.alg

By default, algorithms are given file names based on the associated member type. However, you may rename it with a customized name using the instructions in "Renaming an algorithm." Algorithms can also be deleted when no longer needed, although each member type must have one "active" algorithm in order for the Hub configuration to be deployed.

**Note:** Simply deleting an algorithm from the Navigator view does not remove it from the server from which it was imported. To delete an algorithm, you must use the Member Types tab. Detailed instructions are provided in Chapter 4, "Configuration editor," on page 89.

### Renaming an algorithm

#### About this task

To rename an algorithm, open the Member Types tab in the Hub configuration editor, and select the desired Member type. On the Algorithms tab, select the algorithm you want to rename in the Name field. Type the desired name, being careful to retain the .alg extension, and press the Enter key when finished.

When complete, the algorithm is renamed in Navigator.

### Editing an algorithm

#### About this task

When you edit an algorithm you can add components, edit component properties, connect components to define their relationship, or delete components. Consider the following key factors when editing an algorithm:

- Multiple algorithms from multiple projects can be open for editing simultaneously.
- Available algorithm components are listed in the Palette window, and the components already part of the algorithm configuration are displayed as icons in the editor.
- Algorithms are edited locally so that no changes are made to the database until they have been validated for integrity.
- Algorithms must be saved and deployed before they are updated in the Hub database.

If the algorithm is changed after running Generate Threshold Analysis Pairs, you can rescore the sample pairs by using the **Initiate > Rescore members in sample pairs file(s)** menu. Refer to “Rescoring members in sample pair files” on page 9.

To edit an algorithm:

**Procedure**

1. **Connect to a Hub to retrieve the desired algorithms.** See “Loading an algorithm” on page 154 for more information.
2. Once imported into IBM Initiate Workbench, algorithms are listed in the Navigator, grouped by entity type.
3. Double-click the algorithm name to open the algorithm for editing in the Algorithm editor. The available algorithm components are listed in the Palette, and the algorithm configuration is represented by icons and connectors in the editor. The solid lines define the connections between components, and the dotted lines separate the components by type.

**Results**

The Palette can be collapsed and expanded using the left arrow ( ) and right arrow ( ) buttons in the top left corner of the palette, which may be useful during editing.

---

**The algorithm editor palette**

The following components are listed in the Palette and can be added, edited, or deleted as needed:

- **Connection** – Components dragged to the editor can be connected with the Connection tool.
- **Comparison Role** – Comparison roles define how a comparison function is used in the algorithm.
- **Query Role** – Query roles define how a query function is used in an algorithm.
- **Bucketing Group** – Bucketing groups define a combination of data elements in a bucket.
- **Attributes** – Information describing particular data for a member—person (i.e., name, date of birth, address, SSN) or object (i.e., company name, part number).
- **Standardizations** – Standardization functions “standardize” or format the incoming source data for comparison, bucketing, and search (query) purposes. This can mean capitalization of all alpha characters, removal of punctuation, anonymous value checks, and data ordering. Once standardized, the data is stored as the comparison components of the derived data and is used in the generation of the bucketing data.
- **Bucketing Functions** – Bucketing functions are used for identifying bucketing data, which identify groups of shared information. For example, buckets may be defined for name (first, last, middle), birth date + last name, address, and Social Security number.
- **Comparison Functions** - Comparison functions are used to compare data elements to each other and determine matches and scores. Most of the functions utilize a weight table to make these determinations. The functions work on the...
standardized data created during the data derivation process. The maximum number of comparison functions in a IBM Initiate Workbench project is 16.

The Select button in the Palette allows you to select and highlight a component in the editor, which is necessary to move the component within the editor or display the component’s properties in the Properties tab.

Note: To utilize the full window size for editing, double-click on any editor tab. The Editor view expands to the full height and width of the IBM Initiate Workbench algorithm editor window. Double-clicking one of the editor tabs again restores the normal dimensions of the view.

Setting top-level algorithm properties

About this task

Each algorithm is associated with a particular member type defined in the Hub configuration along with other properties which make it unique to the Master Data Engine instance.

Procedure

1. To display the top-level properties of an algorithm, select any white-space area of the editor, or anywhere in the header section, such as Attributes.
2. Select the Properties tab to display the top-level properties for the algorithm.

Results

See “Top-level algorithm properties.”

Top-level algorithm properties

This section contains a list of the property names and descriptions that you can set for a top-level algorithm function. Refer to this list when adding or editing top-level algorithm functions.

Algorithm tab

- **Derivation code** – defines the derivation code associated with the member type (read-only).
- **Member type** – defines the member type with which the algorithm is associated. This field is editable, allowing the algorithm to be associated with a different member type (read-only).

Thresholds and Weight Properties tab

The following properties are editable. To view weight properties, you must first select an entity type, since weight properties are associated with entity types.

- **Attribute matched pair percentage threshold (wgtNRM)** – defines the threshold for the third filter used in comparison.
- **Attribute matched pair threshold (wgtABS)** – defines the threshold for the second filter used in attribute comparison.
- **Convergence threshold (wgtCNV)** – defines the tolerance for weight generation conversion
- **Data quality percentage for initial weight estimates (wgtQOD)** – defines the matched-set error rate.
- **False negative rate (wgtFNR)** – defines the false negative rate used to compute the Clerical Review and Auto-Link thresholds.
- **False positive rate (wgtFPR)** – defines the false positive rate used to compute the Clerical Review and Auto-Link thresholds.
- **Matched pair threshold (wgtMAT)** – defines the threshold for the first filter used in comparison.
- **Minimum attribute count (wgtFLR)** – defines a lower bound on attribute-value frequency count.

**Note:** For additional information on weight generation, refer to the *IBM Initiate Master Data Service Engine Installation Guide*.

The following describes how to modify an algorithm by adding, modifying, or deleting algorithm components.

**Adding or deleting an attribute**

**Procedure**
1. From the Palette, click **Attributes** to expand the list of attributes available to the algorithm. Attributes define the properties or fields for a data element, and are filtered by the algorithm's member type.
2. Select an attribute from the list.
3. Click in the Attributes column in the editor to drop the selected attribute. To de-select an attribute, click the white space in the editor. A specific attribute can only be added once in an algorithm instance. The editor does not allow you to proceed if you attempt to add an attribute that already exists in the algorithm. There are instances where you require an attribute in a comparison which is already part of a bucketing group, or is included in a query. To solve this, you can connect an attribute to more than one standardization function.
4. To delete an attribute, right-click its icon in the editor and choose **Delete**.

**Viewing attribute properties**

**About this task**

Attribute properties are read-only and cannot be modified.

To view a property for an attribute, select its icon in the Attributes column.

**Procedure**
1. Select the Properties tab to display the properties and values for the attribute.
2. Click the property name or row to view the Value as needed:
   - Description – describes the attribute (read-only)
   - Type – segment code for the attribute (read-only)

**Adding or deleting a standardization function**

**About this task**

The Standardization functions “standardize” or format the incoming source data for comparison, bucketing, and search (query) purposes. This can mean capitalization of all alpha characters, removal of punctuation, anonymous value checks, and data ordering. Once standardized, the data is stored as the comparison components of the derived data and is used in the generation of the bucketing data.
Standardized data is not stored in the Hub database and therefore does not change the member data. For example, a phone number may be entered into a source as 232-123-4567. While the standardization routine may strip the dashes and the area code and format the number as 1234567, the number stored in the Initiate database remains 232-123-4567.

Procedure
1. From the Palette, click Standardizations to expand the list of available standardization functions.
2. Scroll through the list and select a standardization function.
3. Click in the Standardization Functions column in the editor to add the selected standardization function.
4. To delete a standardization function, right-click its icon in the editor and choose Delete.

Setting standardization function properties
About this task
Most properties can be set in the properties view, however some can also be set by right-clicking the standardization icon in the Algorithm editor pane and selecting the desired setting from the context menu.

Procedure
1. To set a property for a standardization function, select its icon in the Standardization Functions column.
2. Select the Properties tab to display the properties and values for the standardization function.
3. Click the property name or row and modify the Value as needed.

Results
See “Standardization function properties.”

Standardization function properties
This section contains a list of the property names and descriptions that you can set for a standardization function. Refer to this list when adding or editing standardization functions.

Anonymous string code
use this if you want to filter out anonymous values (optional). To add new values to the drop-down list, use the Strings tab in the project's Hub Configuration tab. Refer to “Strings” on page 132 for details.

Derivation arguments
contains arguments for data derivation. If multiple arguments are required within the property string, the arguments must be comma-delimited. The equal sign and period characters are allowed in the string.

Description
describes the use of standardization function within the algorithm (optional).

Equivalent string code
use this when you want to filter based on allowable values (optional). To add new values to the drop-down list, use the Strings tab in the project's Hub Configuration tab. Refer to “Strings” on page 132 for details.
Field arguments
0, 1, or more arguments passed to the function (required). Click the ... (ellipsis) button in the Value field (or right-click the standardizations icon and select the Edit Field Arguments option from the context menu) to open the Field Arguments dialog, which displays the available values. (A connector from an Attribute is required.) The Field Arguments dialog enables you to select the desired Fields. The Number of fields column dictates which fields are passed. For example, if you want to pass the onmsuffix to the PXNM function, you must also choose onmprefix, onmmiddle, onmfirst, onmlast fields. If you unselect onmprefix, for example, onmsuffix will not be passed.

Primary standardization role
defines the primary standardization role of the output for standardization functions (required).

Primary standardization role label
contains a description of the associated stdRole. This field can be modified. Up to 32 characters are allowed. While assignment of the stdRole is optional, the label is used by the Anonymous Value Utility during anonymous value selection to assist in the identification of the associated standardization role. If it is not specified, the Utility uses the numeric standardization role identifier.

Record status filter
defines if only active “A”, or both active and inactive “AI” data elements are used (required).

Standardization function code
identifies the standardization function (read-only).

Type
visible when the Use classic functions option is disabled (see “Classic vs. composite algorithm function names” on page 153), determines which standardization function code is used. Not all standardization functions have the type property.

Standardization function with two standardization roles
Some standardization functions, such as BXNM, support two standardization roles. The IBM Initiate Workbench algorithm editor detects this by checking the stdtype2 field for the standardization function’s definition in the engine-metatadatadata.xml file in the project. If stdtype2 is set to non-NULL, the Secondary standardization role property is displayed in the Properties tab. If two standardization roles are supported by the standardization function, the connection from the standardization function to a comparison role or query role will have a property named “Standardization role” displayed in the Properties tab when the connection is selected. The Standardization role property is not visible for those connections where the standardization function has stdtype2 set to NULL (or an empty string).

Note: This is the only case in the editor in which the connection between two algorithm components has a settable property.

To see this in action, we use the Provider algorithm and select the BXNM standardization function.

In the Properties view you see that Secondary standardization role has been added as a property, since BXNM supports two standardization roles. If Secondary standardization role is used (i.e., the standardization function has two connections), this property must be set to a non-zero value:
Select the connector between BXNM and one of its comparison roles (or query roles):

**Standardization role**
- requires selection of either Primary standardization role or Secondary standardization role.

A standardization function which supports two stdRoles can be connected to any combination of one or two comparison and/or query roles. If it has two connections, one connection must have its Standardization role property set to “stdRole1” and the other must be set to “stdRole2”. If the standardization function is connected to only one comparison role or query role (e.g., the second Standardization role is not used), the connection’s Standardization role property must be set to “stdRole1”. Validation rules verify the proper assignment of stdRole1 and stdRole2 when the standardization function supports two roles. Refer to [Weight validation rules](#) on page 180.

**Minimum weight frequency**
- Defines the minimum number of attribute values required in order to be included in the mpi_strfreq table generated by the Generate Frequency Stats (mpxfreq) utility. For example, if “Minimum weight frequency” is 10, only values that occur 10 or more times are listed in the table. In general, the larger the attribute population, the larger this value should be. This property is available for several connections.

**Record status filter**
- defines whether only active (“A”) or both active and inactive (“AI”) data elements are used (required). This property is available for several connections.

**Adding or deleting a comparison role**

**About this task**

Comparison roles are used to define how a comparison function is used in the algorithm.

**Procedure**

1. From the Palette, click **Comparison Role** and drag it to the Comparison and Query Roles column. To de-select the comparison role, click the white space in the editor.
2. To delete a comparison role, right-click its icon in the editor and choose **Delete**.

**Setting comparison role properties**

**Procedure**

1. To set a property for a comparison role, select a comparison role icon in the Comparison and Query Role column.
2. Select the Properties tab to display the properties and values for the comparison role.
3. Click the property name or row and modify the Value as needed.

- **Comparison role**
  - a number used to identity the comparison role (read-only).

- **Description**
  - describes the use of the comparison role within the algorithm (optional).

- **Label**
  - helps identity the comparison role; used as an alternative to the comparison role number if specified (optional).
Adding or deleting a query role

About this task

Query roles are used to define how a query function is used in an algorithm.

Procedure

1. From the Palette, click Query Role and drag it to the Comparison and Query Roles column. To deselect the query role, click the white space in the editor.
2. To delete a query role, right-click its icon in the editor and choose Delete.

Setting query role properties

Procedure

1. To set a property for a query role, select a query role icon in the Comparison and Query Role column.
2. Select the Properties tab to display the properties and values for the query role.
3. Click the property name or row and modify the Value as needed.

   - **Description**
     describes the use of the query role within the algorithm (optional).

   - **Label**
     helps identify the query role; used as an alternative to the query role number if specified (optional).

   - **Query role**
     a number used to identify the query role (read-only).

Adding or deleting bucketing functions

About this task

Bucketing functions are used for identifying bucketing data, which identify groups of shared information. For example, buckets may be defined for name (first, last, middle), birth date + last name, address, and Social Security number.

Procedure

1. From the Palette, click Bucketing Functions to expand the list of available bucketing functions.
2. Scroll through the list and select a bucketing function.
3. Drag and drop the selected bucketing function to the Bucketing and Comparison Functions column in the editor.
4. To delete a bucketing function, right-click its icon in the editor and choose Delete.

Bucketing function properties

This section contains a list of the property names and descriptions that you can set for a bucketing function. Refer to this list when adding or editing bucketing functions.

- **Anonymous string code** – use this if you want to filter out anonymous values (optional). To add new values to the drop-down list, use the Strings tab in the project’s Hub Configuration tab. Refer to “Strings” on page 132 for details.
- **Bucketing function code** – identifies the bucketing function (read-only)
- **Description** – describes the use of the bucketing function within the algorithm (optional)
• **Derivation arguments** – contains arguments for data derivation. If multiple arguments are required within the property string, the arguments must be comma-delimited. The equal sign and period characters are allowed in the string.

• **Equivalent string code** – use this when you want to filter based on allowable values (optional). To add new values to the drop-down list, use the Strings tab in the project's Hub Configuration tab. Refer to "Strings" on page 132 for details.

• **Generation function code** – defines the type of transformation applied to the data (required)

• **Maximum tokens** – indicates the maximum number of bucketing tokens that can be used within the same bucket. Must be greater than 0 and greater than or equal to Minimum tokens. For single-token attributes, this should always be 1 by definition.

• **Minimum tokens** – indicates the minimum number of bucketing tokens that can be used within the same bucket. Set this to 0 to indicate the attribute is optional within the bucket. When it is greater than 0, the attribute must be included in the bucket.

• **Type** – visible when the Use classic functions option is disabled (see "Classic vs. composite algorithm function names" on page 153), determines which bucketing function code is used. Not all bucketing functions have a type property.

### Setting bucketing function properties

#### About this task

Most properties can be set in the properties view, however some can also be set by right-clicking the standardization icon in the Algorithm editor pane and selecting the desired setting from the context menu.

#### Procedure

1. To set a property for a bucketing function, select its icon in the Bucketing and Comparison Functions column.
2. Select the Properties tab to display the properties and values for the bucketing function.
3. Click the property name or row and modify the Value as needed.

   **Note:** Some properties are read-only and cannot be modified.

### Adding or deleting comparison functions

#### About this task

Comparison functions are used to compare data elements to each other and determine matches and scores. Most of the functions utilize a weight table to make these determinations. The functions work on the standardized data created during the data derivation process. The maximum number of comparison functions in a IBM Initiate Workbench project is 16.

The creation of bucket values is a two step process. First a bucketing function is applied to the compare value that was generated by the standardization function. The output of these functions is then modified to produce the bucket values. The nature of this second transformation is determined by the selection of generation function.
**Procedure**

1. From the Palette, click **Comparison Functions** to expand the list of available comparison functions.
2. Scroll through the list and select a comparison function.
3. Drag and drop the selected comparison function to the Bucketing and Comparison Functions column in the editor.
4. To delete a comparison function, right-click its icon and choose **Delete**.

**Comparison function properties**

This section contains a list of the property names and descriptions that you can set for a comparison function. Refer to this list when adding or editing comparison functions.

**Comparison arguments**
contains arguments for comparison. If multiple arguments are required within the property string, the arguments must be comma-delimited alphanumeric sequences. The equal sign and period characters are allowed in the string.

**Comparison function code**
defines the comparison function (read-only).

**Comparison group**
allows two or more statistically-related attributes that are compared separately to be grouped together for final scoring. An example would be Address and Zip code, which represent the same location feature. Only the maximum of these attributes contributes towards scoring, making sure related attributes are not over-weighted. The default group number is 0, which means that every comparison spec is in its own group.

**Comparison mode**
enables the algorithm usage type to be specified for the individual comparison functions employed in the algorithm. Algorithm usage types are: searching, matching and linking (default), matching and linking only, or searching only. The icon displayed depends on the comparison mode value:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Icon 1" /></td>
<td>Use for match, link and search</td>
</tr>
<tr>
<td><img src="image2.png" alt="Icon 2" /></td>
<td>Use for match and link only</td>
</tr>
<tr>
<td><img src="image3.png" alt="Icon 3" /></td>
<td>Use for search only</td>
</tr>
</tbody>
</table>

**Comparison specification code**
a mnemonic used by the weight generation process in defining the wgtCode values in the mpi_wgthead table (required).

**Description**
describes the use of the comparison function within the algorithm (optional).
**Entity Type**
identifies the entity type for which this comparison is valid.

**Equivalent string code**
use this when you want to filter based on allowable values (optional). To add new values to the drop-down list, use the Strings tab in the project’s Hub Configuration tab. Refer to “Strings” on page 132 for details.

**Enable Review Identifier Task**
indicates a review identifier, either “Yes” or “No” (required).

**Kind**
visible when the Use classic functions option is disabled (see “Classic vs. composite algorithm function names” on page 153). Along with Type, the Kind determines which edit distance function is used. This property is currently specific to edit distance functions.

**Type**
visible when the Use classic functions option is disabled (see “Classic vs. composite algorithm function names” on page 153). Determines which comparison function code is used. Not all comparison functions have a type property.

**Weight table percentage cutoff**
defines a cut-off for the weight table. When generating the sval or nval weight table, typically only the most common values are listed. This parameter defines the cumulative percentage of the listed values that should be contained in the weight tables.

### Setting comparison function properties

**About this task**
Most properties can be set in the properties view, however some can also be set by right-clicking the comparison function icon in the Algorithm editor pane and selecting the desired setting from the context menu.

**Procedure**
1. To set a property for a comparison function, select its icon in the Bucketing and Comparison Functions column.
2. Select the Properties tab to display the properties and values for the comparison function.
3. Click the property name or row and modify the Value as needed.

**Note:** Some properties are read-only and cannot be modified.

### Adding or deleting bucketing groups

**About this task**
Bucketing groups define a combination of data elements in a bucket.

**Procedure**
1. From the Palette, click **Bucketing Group** to create a new bucketing group.
2. Drag and drop the Bucketing Group to the Bucketing Groups column in the editor.
3. To delete a bucketing group, right-click its icon and choose **Delete**.
**Bucketing group properties**

- **Bucketing role** – a designation given for the bucket group within an algorithm (required).
- **Derivation group** – a designation to provide a finer granularity for the matching/searching within a single bucketing role (read-only).
- **Description** – describes the use of the bucketing group within the algorithm editor
- **Label** – label used to help identify the bucketing group; used as an alternative to the Derivation group number if specified (optional)
- **Maximum Bucket Size** – maximum size for buckets (required). If "Maximum bucket size" is greater than 0, "Maximum attribute tokens" must be greater than “Minimum attribute tokens.”
- **Maximum attribute tokens** – maximum number of tokens per bucket (should be between 1 and 6)
- **Minimum attribute tokens** – minimum number of tokens per bucket (should be between 0 and 6)

**Setting bucketing group properties**

**Procedure**

1. To set a property for a bucketing group, select its icon in the Bucketing Groups column.
2. Select the **Properties** tab to display the properties and values for the bucketing group.
3. Click the property name or row and modify the Value as needed.

   **Note:** Some properties are read-only and cannot be modified.

**Connecting a component**

**About this task**

Component connections are used to link two components to each other. In the editor, algorithm components are always connected from left to right. For example, to connect an attribute to a standardization function in the editor, the attribute must be selected as the start of the connection, and the standardization function must be selected as the end of the connection.

The editor also controls what components can be connected to other components, restricting connections between components to the adjacent columns. This prevents, for example, a standardization function from being connected to a bucketing function, an attribute from being connected to a comparison role, et cetera.

**Procedure**

1. Click **Connection** from the Palette.
2. Click a component icon from which to start a connection line.
3. Draw a connection line by dragging the line or simply clicking another component in the adjacent column to the right. If an attempted connection is not valid, the line cannot be drawn.
4. Click **Select** to deselect the connection button.
Highlighting associated components

About this task

It is sometimes useful to see which algorithm components are connected to a particular component. By selecting a component in the editor, the other components to which it is connected, either directly or indirectly, as an input or output will be highlighted.

Adding tooltips and labels

About this task

All algorithm components have a description property which is initially empty. You can add a description to each component which will appear as a tooltip when the mouse cursor is hovered over the component in the editor.

As an alternative to displaying the numbers assigned to comparison roles, query roles, and bucketing groups, you may want to use descriptive labels. If specified, a label appears beneath the role or group in the editor. (This is different from a description, which is displayed as a tooltip when the mouse cursor is hovered over an algorithm component.)

To add tooltips

Procedure

1. Select a component in the editor by clicking on it.
2. In the Properties view, highlight Description.
3. Type the description directly in the Value column.

Results

Now when your mouse cursor hovers over the component, the description is shown.

Note: The Description field for Attribute items cannot be edited in the Algorithm Editor. Use the Configuration Editor to make changes to the Attribute description.

To add labels

Procedure

1. Double-click the object in the editor.
2. Type the desired label text in the text box that opens. The new label text appears.

Weight generation parameters

The following parameters can be set via the IBM Initiate Workbench algorithm editor and should be specified prior to generating weights. Refer to Chapter 3, “Jobs and job sets,” on page 21 for detailed information on the weight generation job. Refer to Chapter 6, “Weight generation,” on page 175 for a list of weight generation steps.

Table 48. Matched pair threshold

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database table</td>
<td>mpi_cmphead</td>
</tr>
</tbody>
</table>
### Table 48. Matched pair threshold (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Used by</strong></td>
<td>Compare Members in Bulk (mpxcomp) – during matched pair generation</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Generates a set of matched pairs that are used to calculate weights throughout the process. The weight generation process runs Compare Members in Bulk (mpxcomp) to find any pairs that score above wgtMAT. All pairs that score below wgtMAT are filtered out.</td>
</tr>
<tr>
<td><strong>Scale</strong></td>
<td>One implied decimal point; 120 is actually 12.0</td>
</tr>
<tr>
<td><strong>Valid range/ Default and Recommended setting</strong></td>
<td>10..200 Default: 100 Recommended setting: 100</td>
</tr>
<tr>
<td><strong>Property location</strong></td>
<td>Thresholds and Weight Properties tab Properties view</td>
</tr>
</tbody>
</table>

### Table 49. Attribute matched pair threshold

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database table</strong></td>
<td>mpi_cmphead</td>
</tr>
<tr>
<td><strong>Used by</strong></td>
<td>Compare Members in Bulk (mpxcomp) – during matched pair generation</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>This is the threshold for the second filter. As weight generation calculates weights for each attribute, it refines the list of matched pairs using all but the current attribute. The pairs must score above wgtABS. If the score is below wgtABS, the pair is filtered out.</td>
</tr>
<tr>
<td><strong>Scale</strong></td>
<td>One implied decimal point; 120 is actually 12.0</td>
</tr>
<tr>
<td><strong>Valid range/ Default and Recommended setting</strong></td>
<td>10..180 Default: 80 Recommended setting: 80</td>
</tr>
<tr>
<td><strong>Property location</strong></td>
<td>Thresholds and Weight Properties tab Properties view</td>
</tr>
</tbody>
</table>

### Table 50. Attribute matched pair percentage threshold

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database table</strong></td>
<td>mpi_cmphead</td>
</tr>
<tr>
<td><strong>Used by</strong></td>
<td>Compare Members in Bulk (mpxcomp) – during matched pair generation</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>After using wgtABS to find matched pairs, this filter is applied. The wgtNRM filter only considers matched pairs where the score is a certain percentage of the exact match score. If this percent falls below wgtNRM, the pair is filtered out.</td>
</tr>
<tr>
<td><strong>Scale</strong></td>
<td>A percent; a value of 95 is actually 95%</td>
</tr>
<tr>
<td><strong>Valid range/ Default and Recommended setting</strong></td>
<td>60..100 Default: 95 Recommended setting: 95</td>
</tr>
<tr>
<td><strong>Property location</strong></td>
<td>Thresholds and Weight Properties tab Properties view</td>
</tr>
</tbody>
</table>
Table 51. Data quality percentage for initial weight estimates

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database table</td>
<td>mpi_cmphead</td>
</tr>
<tr>
<td>Used by</td>
<td>mpxwgt – during initial weight generation (without matched set)</td>
</tr>
<tr>
<td>Description</td>
<td>When computing the initial weights using unmatched weights, this parameter defines the matched set error rate, or the percentage of matched attributes that disagree.</td>
</tr>
<tr>
<td>Scale</td>
<td>A percent; a value of 5 is actually 5%</td>
</tr>
</tbody>
</table>
| Valid range/ Default and Recommended setting | 1..20  
Default = 5  
Recommended setting: 5 |
| Property location  | Thresholds and Weight Properties tab Properties view   |

Table 52. Minimum attribute count

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database table</td>
<td>mpi_cmphead</td>
</tr>
<tr>
<td>Used by</td>
<td>mpxwgt</td>
</tr>
<tr>
<td>Description</td>
<td>This property (wgtFLR) defines a lower bound on attribute value frequency counts. When the count is less than the minimum attribute count, it is raised to equal the minimum attribute count.</td>
</tr>
<tr>
<td>Scale</td>
<td>No scaling; a value of 20 means 20</td>
</tr>
</tbody>
</table>
| Valid range/ Default and Recommended setting | >0  
Default = 5  
Recommended setting: 5 |
| Property location  | Thresholds and Weight Properties tab Properties view   |

Table 53. Convergence threshold

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database table</td>
<td>mpi_cmphead</td>
</tr>
<tr>
<td>Used by</td>
<td>mpxconv</td>
</tr>
<tr>
<td>Description</td>
<td>Provides the tolerance for weight generation convergence. The weight generation process performs multiple iterations until weights from the latest run match weights from the previous run within the points specified in the convergence threshold. For example if a value of 50 is supplied, then the weights converge (wgtCNV) for the two previous iterations when no two weights differ by more than 0.50. Iteration stops when convergence is reached.</td>
</tr>
<tr>
<td>Scale</td>
<td>Two implied decimal points; a value of 50, is actually 0.50</td>
</tr>
</tbody>
</table>
| Valid range/ Default and Recommended setting | 1..100  
Default = 20  
Recommended setting: 20 |
| Property location  | Thresholds and Weight Properties tab Properties view   |

Table 54. False negative rate

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database table</td>
<td>mpi_cmphead</td>
</tr>
<tr>
<td>Used by</td>
<td>mpxdist</td>
</tr>
</tbody>
</table>
Table 54. False negative rate (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>False Negative Rate (wgtFNR). Used by mpxdist to compute the Clerical Review and Auto-link thresholds based on desired false negative rate.</td>
</tr>
<tr>
<td>Scale</td>
<td></td>
</tr>
<tr>
<td>Valid range/ Default and Recommended setting</td>
<td>1..100</td>
</tr>
<tr>
<td>Recommended setting</td>
<td>100</td>
</tr>
<tr>
<td>Property location</td>
<td>Thresholds and Weight Properties tab Properties view</td>
</tr>
</tbody>
</table>

Table 55. False positive rate

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database table</td>
<td>mpi_cmphead</td>
</tr>
<tr>
<td>Used by</td>
<td>mpxdist</td>
</tr>
<tr>
<td>Description</td>
<td>False Positive Rate (wgtFPR). Used by mpxdist to compute Auto-link thresholds based on desired false positive rate.</td>
</tr>
<tr>
<td>Scale</td>
<td></td>
</tr>
<tr>
<td>Valid range/ Default and Recommended setting</td>
<td>&gt;0</td>
</tr>
<tr>
<td>Recommended setting</td>
<td>100000</td>
</tr>
<tr>
<td>Property location</td>
<td>Thresholds and Weight Properties tab Properties view</td>
</tr>
</tbody>
</table>

Table 56. Weight table percentage cut-off

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database table</td>
<td>mpi_cmpspec</td>
</tr>
<tr>
<td>Used by</td>
<td>mpxwgt</td>
</tr>
<tr>
<td>Description</td>
<td>Defines a cut-off for the weight table. When generating the sval or nval weight table, typically only the most common values are listed. wgtCUT defines the cumulative percentage of the listed values that should be contained in the weight tables.</td>
</tr>
<tr>
<td>Scale</td>
<td>Percentage; a value of 80 means 80%</td>
</tr>
<tr>
<td>Valid range/ Default and Recommended setting</td>
<td>1..100</td>
</tr>
<tr>
<td>Default = 80</td>
<td>Default = 80</td>
</tr>
<tr>
<td>Recommended setting</td>
<td>80</td>
</tr>
<tr>
<td>Property location</td>
<td>Algorithm tab Properties view (comparison function)</td>
</tr>
</tbody>
</table>

Table 57. Minimum weight frequency

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database table</td>
<td>mpi_dvdxcmp</td>
</tr>
<tr>
<td>Used by</td>
<td>Generate Frequency Stats (mpxfreq) – in wgtmode</td>
</tr>
<tr>
<td>Description</td>
<td>Defines the minimum frequency of values to be listed in the strfreq table. If minWgtFreq is 10, only values that occur 10 or more times are listed in the table. In general, the larger the attribute population, the larger the minWgtFreq number should be.</td>
</tr>
<tr>
<td>Scale</td>
<td>No scaling; a value of 10 means 10</td>
</tr>
</tbody>
</table>
Table 57. Minimum weight frequency (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid range/ Default</td>
<td>&gt;=0&lt;br&gt;Default = 20&lt;br&gt;Recommended setting: 20 for all attributes other than dates. Dates should be set at 1.</td>
</tr>
<tr>
<td>and Recommended setting</td>
<td></td>
</tr>
<tr>
<td>Property location</td>
<td>Algorithm tab Properties view (comparison role)</td>
</tr>
</tbody>
</table>

The sample screens that follow show the weight generation parameters in the Thresholds and Weight Properties view in the algorithm editor.

The weight generation utility uses the binary files `mpi_membktd.NNN` and `mpi_memcmpd.NNN`. These files can be generated using a number of utilities, depending on the state of your system.

Table 58. Weight generation utilities used for various system states

<table>
<thead>
<tr>
<th>Utility</th>
<th>System state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Derive Data and Create UNLs (mpxdata)</td>
<td>Your member data is not loaded in the database, but exists in a flat load file.</td>
</tr>
<tr>
<td>Prepare Binary Files (mpxprep)</td>
<td>A) Your member data, including your cmpd and bkted data, is loaded in the database, B) your mpi_memcmpd and mpi_membktd tables are current, and C) no changes have been made to the algorithm since mpi_memcmpd and mpi_membktd have been generated.</td>
</tr>
<tr>
<td>Derive Data from Hub (mpxredvd)</td>
<td>Your member data is loaded in the Initiate database. Since, this utility generates the cmpd and bkted data, it may be used if mpi_memcmpd and mpi_membktd are not populated or are out of date.</td>
</tr>
<tr>
<td>Derive Data from UNLs (mpxfsdvd)</td>
<td>Your member data exists as .unl files.</td>
</tr>
</tbody>
</table>

The Jobs feature contains options for running any of the above utilities or using membktd.NNN and mpi_memcmpd.NNN files that have already been generated. For more information on these utilities, refer to [Chapter 3, “Jobs and job sets,” on page 21](#).

**Weight generation properties**

The Generate Weights utility is built upon the madconfig Ant framework used to configure other Initiate software components. The weight generation utility can now be executed through IBM Initiate Workbench, but all of the command line arguments from the madconfig prompts are still available.

As with all madconfig targets, unattended execution can be scripted out by using the `-recordfile` and `-propertyfile` options. The following is a summary of the important madconfig properties that are used by the generate_weights target. All of the properties can be overridden in a properties file that is passed in with the `propertyfile` madconfig command line option. Refer to the IBM Initiate Master Data Service Engine Installation Guide for details about using madconfig.
### Table 59. Weight generation properties with prompts

<table>
<thead>
<tr>
<th>Property</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mad.db.dsn</td>
<td>N/A</td>
</tr>
<tr>
<td>mad.db.type</td>
<td>N/A</td>
</tr>
<tr>
<td>mad.db.user</td>
<td>N/A</td>
</tr>
<tr>
<td>mad.db.password</td>
<td>N/A</td>
</tr>
<tr>
<td>mad.wgtgen.basedir</td>
<td>${mad.root.dir}/zwgts</td>
</tr>
<tr>
<td>mad.wgtgen.entity.type</td>
<td>N/A</td>
</tr>
<tr>
<td>mad.wgtgen.step</td>
<td>N/A</td>
</tr>
<tr>
<td>mad.wgtgen.continue</td>
<td>N/A</td>
</tr>
<tr>
<td>mad.wgtgen.dvddata.option</td>
<td>N/A</td>
</tr>
<tr>
<td>mad.wgtgen.member.unl.dir</td>
<td>N/A</td>
</tr>
<tr>
<td>mad.wgtgen..config.file</td>
<td>N/A</td>
</tr>
<tr>
<td>mad.wgtgen..input.file</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Table 60. Weight generation properties without prompts

<table>
<thead>
<tr>
<th>Property</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mad.wgtgen.log.file</td>
<td>${mad.wgtgen.basedir}/wgtgen.log</td>
</tr>
<tr>
<td>mad.wgtgen.bxm.dir</td>
<td>${mad.wgtgen.basedir}/bxm</td>
</tr>
<tr>
<td>mad.wgtgen.unl.dir</td>
<td>${mad.wgtgen.basedir}/unl</td>
</tr>
<tr>
<td>mad.wgtgen.frq.dir</td>
<td>${mad.wgtgen.basedir}/frq</td>
</tr>
<tr>
<td>mad.wgtgen.upairs.dir</td>
<td>${mad.wgtgen.basedir}/upairs</td>
</tr>
<tr>
<td>mad.wgtgen.usamps.dir</td>
<td>${mad.wgtgen.basedir}/usamps</td>
</tr>
<tr>
<td>mad.wgtgen.mpairs.dir</td>
<td>${mad.wgtgen.basedir}/mpairs</td>
</tr>
<tr>
<td>mad.wgtgen.msamps.dir</td>
<td>${mad.wgtgen.basedir}/usamps</td>
</tr>
<tr>
<td>mad.wgtgen.run.dir</td>
<td>${mad.wgtgen.basedir}/run</td>
</tr>
<tr>
<td>mad.wgtgen.cmpmode</td>
<td>3</td>
</tr>
<tr>
<td>mad.wgtgen.threads</td>
<td>1</td>
</tr>
<tr>
<td>mad.wgtgen.max.iterations</td>
<td>5</td>
</tr>
<tr>
<td>mad.wgtgen.num.parts.1</td>
<td>20</td>
</tr>
<tr>
<td>mad.wgtgen.num.parts.2</td>
<td>5</td>
</tr>
<tr>
<td>mad.wgtgen.num.parts.3</td>
<td>5</td>
</tr>
<tr>
<td>mad.wgtgen.num.parts.frq</td>
<td>10</td>
</tr>
<tr>
<td>mad.wgtgen.num.parts.max</td>
<td>10</td>
</tr>
<tr>
<td>mad.wgtgen.audrecno</td>
<td>99</td>
</tr>
<tr>
<td>mad.wgtgen.upair.count</td>
<td>3000000</td>
</tr>
<tr>
<td>mad.wgtgen.report.records</td>
<td>10000</td>
</tr>
<tr>
<td>mad.wgtgen.max.bucket.size</td>
<td>1000</td>
</tr>
<tr>
<td>mad.wgtgen.min.weight</td>
<td>4.0</td>
</tr>
<tr>
<td>mad.wgtgen.blk.size</td>
<td>100000</td>
</tr>
<tr>
<td>mad.wgtgen.unl.segments.</td>
<td>memhead, membktl, memcmpd, memname, memaddr, memphone, memattr, memident</td>
</tr>
</tbody>
</table>
Setting CR and AL thresholds

About this task

In the following steps, you create the mapping of the two sources being compared and define the Clerical Review and Auto-link scores used in the comparison. This information is stored in MPI_srcxsrc. Perform these steps for each source pair you want to compare.

Setting thresholds manually

Procedure

1. Click the Thresholds and Weight Properties tab.
2. Select the Entity Type. If you are using more than one type, you will have to add comparison scores to each source pair separately.
3. Select a source pair from the “Thresholds for selected Entity Type” list. You can [Shift]- or [Ctrl]-click to multi-select rows and change properties for more than one source pair at a time.
4. Select the desired Active setting. When this is false, threshold values will not be added to mpi_srcxsrc for that source/source combination.
5. Type the Clerical Review Threshold in the appropriate column. The software assumes a decimal; for example if you enter 8, the software converts it to 8.0. Records that score above this threshold and below the Auto-Link threshold are placed into tasks.
6. Type the Autolink Threshold. Again, the software assumes a decimal. All records scoring above this threshold are automatically linked and assigned common Enterprise ID numbers.

Using the Threshold Calculator

About this task

The threshold calculator enables you to use sample data from your Hub database to calculate the appropriate Clerical Review and Autolink thresholds based on a False Negative Rate (FNR), False Positive Rate (FPR) and Task count you provide. You can then use these values for selected (or all) source pairs.

The thresholds are calculated using either an estimated FPR or a statistical FPR based on evaluated sample pair data. The statistical option requires that you first run the Generate Threshold Analysis Pairs job, and then evaluate the sample pairs created by that utility. Refer to “Database Tools jobs” on page 78.

Note: Run the Compare Members in Bulk (mpxcomp) and Link Entities (mpxlink) utilities prior to running the Generate Threshold Analysis Pairs utility.
Procedure

1. Click the **Threshold Calculator** button. The Threshold Calculator wizard opens.

2. Use the top **Browse** button to provide the name of the dsamp file containing the sample data to work with. This file was generated during the last step in the weight generation process and copied to the project's **Weights\Entity_Type** folder when you performed the **Get job results** function on the Generate Weights job.

3. Indicate whether you wish to use an **Estimated** FPR, or **Statistical** FPR based on evaluated sample pair data.
   - If you choose the Estimated FPR option, data appears in the Error Rates graph only, and the task estimate count will be empty.
   - If you choose the Statistical option, browse to the sample pairs .xls file(s) generated by the Generate Threshold Analysis Pairs job. The file(s) is located in IBM Initiate Workbench's **workspace\project_name** directory by default. If multiple sample pair files were generated and evaluated, they must be in the same directory and uploaded at the same time.

   **Note:** If an error message ("Unable to create a statistical model from evaluated data given. Please check the data for inaccuracies.") appears when you attempt to select the sample pairs .xls file, verify that Yes, No or Maybe has been selected in the “Same?” column for at least one member in all pairs. Refer to "Generate Threshold Analysis Pairs” on page 30.

4. Choose **Enable subset calculations** if the analysis is being done on a subset of the final database rather than on the final database. When this option is enabled, you should also indicate the approximate **Number of members in final database** for best results.

5. Click **Next**.

6. On the Threshold and Error Rate Calculations page, set the values for each field as desired, clicking the **Recalculate** button as needed to perform the calculations. Each tab provides a graph of the sample data for
   - Error rates
   - Sample pair FPR
   - Sample pair percent same/not same
   - Score distribution
   - Task estimates
   
   You can use this calculator to set a clerical review threshold and autolink threshold to get an estimate on the false positive rate, false negative rate and estimated number of tasks.

7. When the FPR, FNR and number of tasks are set appropriately for your data, click **Finish**. The Save Thresholds dialog opens.

8. Select the source combinations for which you wish to apply the calculated Clerical Review Threshold and Autolink Threshold values. The **Select All**, **Deselect All**, **Select Same** and **Select Different** buttons provide a quick way to select combinations. You can also Shift- or Ctrl-click to multi-select source combinations and check or uncheck their select boxes. Click **Save** to finish.
Chapter 6. Weight generation

The Hub enables you to generate weights for attributes associated with a particular entity type by setting values via the IBM Initiate Workbench algorithm editor and then running a job set which includes the Generate Weights utility. Consult Chapter 3, “Jobs and job sets,” on page 21 for instructions on how to use a job set to run utilities.

Before beginning the weight generation process, you must have the following items in place.

- A configured algorithm for the entity type, which can be done through the algorithm editor. Refer to the tables in “Weight generation parameters” on page 167 for complete descriptions of the tables that the IBM Initiate Workbench algorithm editor reads and writes for weight generation.
- Parameter values for weight generation, which are set in the Thresholds and Weight Properties tab in the algorithm editor and persisted as part of the algorithm configuration (see “Weight generation parameters” on page 167). Refer to “Setting CR and AL thresholds” on page 173.
- A set of member data and the following derived data binary files: mpx_memcmpd.bin and mpx_membktd.bin. Note that derived data files can be generated during the weight generation process.
- You must also set up and populate any anonymous value and equivalent value tables used by the standardization functions (mpi_stranon). This is very important, as unfiltered anonymous values can distort the weight values. Use the Anonymous Value Utility to generate values. (Consult “Anonymous Value Utility” on page 142 for instructions on using the Anonymous Value Utility.)

The following provides an overview of the process for obtaining accurate weights. Instructions on running utilities are found in Chapter 3, “Jobs and job sets,” on page 21.

1. Generate derived data. This is done by running one of the following utilities: Derive Data and Create UNLs (mpxdata), Prepare Binary Files (mpxprep), Derive Data from UNLs (mpxsfdvd) or Derive Data from Hub (mpxredvd). The differences between these utilities are discussed in Chapter 3, “Jobs and job sets,” on page 21.

2. Generate your frequency files. This is done by running the Generate Frequency Stats (mpxfreq) utility using the Generate raw frequency table output option. The output of mpxfreq must then be copied into the project from the Hub via the Jobs view by running the “Get job results” action on the successful Generate Frequency Stats (mpxfreq) job.

3. Run the Anonymous Value tool to select anonymous values from the data generated in the previous step. Consult “Anonymous Value Utility” on page 142 for instructions on how to run the Anonymous Value Utility. Upon saving the selections, the anonymous string tables are updated in the project.

4. Deploy the configuration to the Hub.

5. Run the Generate Weights utility. See “Generating weights” on page 176 for more information.

6. Run the Validate Weights utility. See “Validating Weights” on page 179 for more information.
7. Retrieve the results of weight generation from the Hub via the Jobs view by running the “Get job results” action on the Generate Weights job. This includes the generated weight tables as well as the mpx_dsamp_<enttype>.txt file, which is used as input to the Threshold Calculator.

8. Run the Compare Members in Bulk (mpxcomp) and Link Entities (mpxlink) utilities as described in “Compare Members in Bulk (mpxcomp)” on page 40 and “Link Entities (mpxlink)” on page 64.

9. Run the Generate Threshold Analysis Pairs jobs “Database Tools jobs” on page 78.

10. Evaluate and adjust the Clerical Review Threshold and Autolink Threshold for each source combination and entity type. Instructions are found in the section titled “Setting CR and AL thresholds” on page 173.

11. Deploy the configuration to the Hub.

---

**Enabling frequency-based bucketing**

**About this task**

The following are optional steps to enable frequency-based bucketing:

**Procedure**

1. To enable frequency-based bucketing on a particular bucketing group, select the bucketing group in the algorithm editor, then set the “Maximum bucket size” property in the Properties view to a value greater than 0, then re-deploy the algorithm to the Master Data Engine.

   **Note:** If “Maximum bucket size” is greater than 0, “Maximum attribute tokens” must be greater than “Minimum attribute tokens.”

2. Generate your frequency files. This is done by running the Generate Frequency Stats (mpxfreq) utility using the Generate frequency tables for frequency-based bucking option. The output of mpxfreq must be copied into the project from the Master Data Engine via the Jobs view by running the Get job results action on the successful Generate Frequency Stats (mpxfreq) job.

3. Deploy the configuration to the Master Data Engine.

**Results**

Remember to regenerate derived data before continuing with weight generation.

---

**Generating weights**

**About this task**

Before generating the weights, verify that all significant anonymous data has been identified and that the mpi_stranon table is populated.

**Procedure**

1. In IBM Initiate Workbench, create a new job set by clicking the Create a new job set button on the toolbar. For details on creating and running a job set, refer to Chapter 3, “Jobs and job sets,” on page 21.

2. Expand the Configuration section and add the Generate Weights job. Click OK.

3. Select the desired Entity Type.
4. On the Steps tab, select the weight generation step you wish to begin with. You can begin at any point in the process (see the step descriptions in “Generate Weights process step descriptions”). You can also indicate whether you want to execute only that step or continue through the remaining steps of the process.

5. Use the checkbox to indicate whether you wish to Execute all remaining steps through the end of the process. Usually this is enabled so that each step in the process is performed in order.

6. Set the remaining properties on the other tabs as needed.

Note: Double check that the “Number of comparison bucket partitions” value on the Performance Tuning tab matches the corresponding property value selected in the utility that was used to generate the derived data. The default value may not be correct.

7. Click Finish.

8. If prompted, enter a valid user name and password. The job begins to run.

9. Once it has finished successfully, expand the Generate Weights job in the Jobs view.

10. Right click the first “child” entry and select Get job results. This will return all of the weight tables.

11. Continue with “Setting CR and AL thresholds” on page 173.

Generate Weights process step descriptions

In the Generate Weights job, Steps tab, you select the weight generation step you wish to begin with. You can begin at any point in the process.

You can also indicate whether you want to execute only that step or continue through the remaining steps of the process. The description of the steps are as follows:

Delete artifacts from previous run
serves as a “clean-up” step. If selected, all directories within your existing weight generation working directory are deleted.

Generate counts for all attribute values
computes value counts and populates the frq directory. The Number of frequency partitions property is used to define the number of folders (parts) that will be created in the frq directory. If you have a large amount of data, the ability to break the data apart can prevent encountering folder size limits. This step uses the “Minimum weight frequency” property (set in the Algorithm editor) to determine for each comparison role the smallest count that will appear in the frequency table.

Generate random pairs of members
generates random pairs of members (MEMRECNOs), populates the upairs directory and produces the file mpx_usamp_entitytype.txt, which is used as input to the Measuring Accuracy Calculator. This file contains the scores created from comparing the members chosen at random. Since no weights have been generated from this process yet, the bootweights are used for this step. After weight generation is complete, use the Compare Members in Bulk (mpxcomp) job to compare the random members and get the usamps file using the FINAL weights. This step also forms bucket files (mpx_bcmktd.NNN). The “Number of random pairs to generate” and “Number of random pairs bucket partitions” properties are used in this step. See “Weight generation properties” on page 171.
If the **Execute all remaining steps through the end of the process option** is enabled, the random pair comparison is done again towards the end of weight generation using FINAL weights. This additional step produces two extra files in the final weight directory: `mpx_usamp_entitytype.txt` and `mpx_usamp_entitytype.bin`, and cannot be run by itself.

**Derive random data by comparing random members**
This step is comprised of two processes. The first process compares random results by applying the comparison algorithm to the random pairs. Then it uses the results from the attribute value counts step to generate initial weights. The Data quality percentage for initial weight estimates parameter is used in the initial weight computation. Additionally, the Weight table percentage cut-off parameter is used to determine the percentage of values that are written to the nval and sval weight tables. Populates the usamp directory.

**Perform matched candidate pairs reduction**
performs matched candidate pairs reduction and populates the mpairs directory. The minimum score used for matching is found in `mpi_cmphead.wgtMAT`. All pairs that score below the Matched pair threshold are filtered out in this step. For each attribute defined in the algorithm, three filters are applied to each candidate pair. All filter settings are defined in `mpi_cmphead` and consist of:

- **Matched pair threshold**
  - if the comparison score is below this setting, the pair is thrown out.

- **Attribute matched pair threshold**
  - the second comparison is performed without using the attribute in question and if the score is less than the Attribute matched pair threshold, the pair is thrown out.

- **Attribute matched pair percentage threshold**
  - the comparison score modulo the attribute is recomputed as a percentage of an exact match and if the percentage is less than the Attribute matched pair percentage threshold, the pair is thrown out.

**Generate matched set, matched statistics and initial weights**
applies comparison algorithm and generate matched set, and populates the msamp directory. If a pair is not thrown out after the filtering process described in the Perform matched candidate pairs reduction option above, the result is counted in the matched results. The matched set is used to compute matched set frequencies and then weights. The comparison uses the following parameters:

- **Minimum attribute count**
  - when the attribute count falls below this setting, the count is set to equal this number.

- **Matched pair threshold**
  - for each pair score that falls below this value, the pair is excluded.

- **Attribute matched pair threshold**
  - for each attribute, if the score modulo the attribute score falls below this parameter, it is excluded.

- **Attribute matched pair percentage threshold**
  - if the score is less than this parameter, the pair is excluded from the matched set.

**Skip last step because of too few attributes**
when the number of attributes is low, you can skip the final (iteration) step.
This checkbox is enabled only if the Generate matched set, matched statistics and initial weights option will be executed.

**Iterate over previous step and check for convergence of weights**

This step re-computes the matched set, frequencies, and then weights, then checks the resulting weights from the latest iteration against weights from the previous iteration for convergence. Following each iteration, Weight Generation writes the results to mpi_wgthead and the weight files (e.g., mpi_wgt#dim, mpi_wgtval, mpi_wgtvaln). When weights converge (latest and previous weights match within the points specified in mpi_cmphead.wgtCNV), the iteration process stops.

**Note:** A new utility, mpxsmooth, is automatically run during the Generate Weights job to smooth weights. For information on the mpxsmooth utility, consult the IBM Initiate Master Data Service Engine Installation Guide.

---

### Validating Weights

**About this task**

Once weights are generated, it is a good idea to validate the weights to verify whether the weights fall into accepted parameters. Should you find that weights need adjustments, you can use the procedures described in “Setting CR and AL thresholds” on page 173.

**Validating weights using a job**

Validate Weights can be run any time after the Generate Weights job, and even within the same job set, if desired.

**About this task**

More detailed information on the Validate Weights job options can be found in Chapter 3, "Jobs and job sets," on page 21.

**Procedure**

1. Create a job set that includes the Validate Weights job.
2. When the Validate Weights job finishes successfully, right click the listing in the Jobs view and select Get Job Results. The results file will identify the weight values that need to be adjusted in a “remediation” tag.

**Validating weights locally**

**About this task**

Alternatively, you can validate weights contained in local copies of weight UNL files.

**Procedure**

1. After the Generate Weights job is completed, right-click the completed job and execute the Get job results action. This action retrieves the weight UNL files from the Hub and saves them in the project locally.
2. From the menu, select Initiate > Validate Local Weights. The Validate Local Weights dialog opens.
3. Select the Project containing the weight UNL files to be validated.
4. Use the Browse button to specify the directory within the project containing the UNL files.
5. Select the desired report file format. Plain text and XML formats are available.
6. If you enable the Open weight validation report after completion option, the report will automatically open in the IBM Initiate Workbench editor once it is created.
7. Click Finish.

Weight validation rules

The following are descriptions of the heuristics or "rules" implemented for Weight Validation. The rules are split into "level 1" and "level 2" categories. A failure of a level 1 rule will generate an error; a failure of a level 2 rule will generate a warning. All weight values have an explicit or implicit decimal point. If a weight value is represented without a decimal, the "real" weight value is calculated by dividing it by 100.0f. Weight values in the report are all converted to their decimal format, i.e., divided by 100.0f. If there are rule conflicts, level 2 rules supersede level 1 rules.

Weight ranges

In general, if a rule uses the word "between," such as "INITIAL_ADJWGT should be between 500 and 800," the specified range is inclusive, i.e., the lower and upper weight values are also valid. If a rule uses the phrase "less than" and/or "more than," these specified lower/upper weight values are exclusive, i.e., they are not valid weight values.

Checkpoints

Checkpoints are instructions provided to help the user manually "fix" the weight values or may suggest changes to the algorithm configuration in order to produce "better" weights on a subsequent run of weight generation. In general, if weights in the dimensional weight tables (1Dim, 2Dim, etc.) generate warnings and/or errors, this usually reflects problems in the data (e.g., not enough data) or in the weight generation utilities. In the latter case, the problems should be reported to the data scientists for analysis.

Level 1 rules

A failure of any of these rules will generate a warning.

Basic Nval rule

For each wgtcode in the nval table:
1. The minimum must be greater than zero.
2. If there is a default agreement weight (numval=-1), it should be greater than or equal to the minimum value.

| CMPID-DOB-YEAR | 1949 | 438 |
| CMPID-DOB-YEAR | 1993 | 440 |
| CMPID-DOB-YEAR | -1   | 466 |

Basic Sval rule

For each wgtcode in the sval table:
Calculate the maximum and the minimum of the exact-match weights (wgtcode is XACT).

Find the discrepancy weight (strval=d) and the default weight (strval=a).
1. The minimum of the exact-match weights must be greater than zero.
2. The discrepancy weight must be less than zero.
3. The default weight must be greater than or equal to the minimum of the exact-match weights.

**Basic 1Dim rule**

For each wgtcode in the file:
1. The weight for wgtidxno=0 should be zero.
2. If the wgtcode is DIST, and cmpfuncno != AXP:
   a. The maximum must be greater than zero, the minimum must be less than zero.
   b. The wgtidxno values should be contiguous beginning with zero.
   c. The wgtvals should be monotonically decreasing with increasing wgtidxno (not including wgtidxno=0).

<table>
<thead>
<tr>
<th>CMPID-SSN-DIST</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPID-SSN-DIST</td>
<td>1</td>
<td>553</td>
</tr>
<tr>
<td>CMPID-SSN-DIST</td>
<td>2</td>
<td>304</td>
</tr>
<tr>
<td>CMPID-SSN-DIST</td>
<td>3</td>
<td>98</td>
</tr>
</tbody>
</table>

**Basic 2Dim rule**

For each wgtcode in the file, this represents a 2-dimensional array of weight values.
- The rows are indexed by wgtidxno1 and the columns have the weights associated with wgtidxno2.
- The maximum value of wgtidxno2 is 15, but often not all 15 positions are used.
- The remaining values are padded with zero.
- The values in the second column going down the page are wgtidxno1.
- The values in each row are wgtidxno2.

<table>
<thead>
<tr>
<th>CMPID-AXP-2DIM</th>
<th>0</th>
<th>0</th>
<th>412</th>
<th>334</th>
<th>256</th>
<th>147</th>
<th>48</th>
<th>-39</th>
<th>-199</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPID-AXP-2DIM</td>
<td>1</td>
<td>316</td>
<td>419</td>
<td>334</td>
<td>334</td>
<td>334</td>
<td>334</td>
<td>334</td>
<td>334</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CMPID-AXP-2DIM</td>
<td>2</td>
<td>297</td>
<td>394</td>
<td>334</td>
<td>334</td>
<td>334</td>
<td>334</td>
<td>334</td>
<td>334</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CMPID-AXP-2DIM</td>
<td>3</td>
<td>247</td>
<td>334</td>
<td>334</td>
<td>334</td>
<td>334</td>
<td>334</td>
<td>334</td>
<td>334</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Find the maximum value where wgtidxno2 has a non-zero weight.
1. This should be the same for all rows (e.g. for all values of wgtidxno1). Call this max2. (In the above example, max2 = 7)
   The weights for wgtidxno1=0 and wgtidxno2=0,..., max2 are analyzed as a 1 dimensional weight table, so
2. The weight for wgtidxno1=wgtidxno2=0 should be zero.
3. Calculate the maximum and minimum weights for this row excluding wgtidxno2=0. The maximum should be greater than zero and the minimum should be less than zero. In the above example, in row 1, max=412>0>-199=min.
4. The weights should be monotonically decreasing with increasing wgtidxno2.
   The weights for wgtidxno2=0 and wgtidxno1=0,..., maxrows are analyzed as a 1 dimensional weight table, so
5. The weight for wgtidxno1=wgtidxno2=0 should be zero.

6. Calculate the maximum and minimum weights for this row excluding wgtidxno1=0. The maximum should be greater than zero and the minimum should be less than zero.

7. The weights should be monotonically decreasing with increasing wgtidxno1.

8. For wgtidxno1 > 0 and for wgtidxno2 > 0, we require that the weights decrease in both dimensions. So:
   - wtgtval(wgtidxno1, wgtidxno2) >= wtgtval(wgtidxno1+1, wgtidxno2)
     (In the above example, 419>394>334...) and
   - wtgtval(wgtidxno1, wgtidxno2) >= wtgtval(wgtidxno1, wgtidxno2+1)
     (In the above example:334>256>147>48>...)

Note that we skip row and col 0 for both cases in step 8 as those cases are handled in step 4 and 7.

**Basic 3Dim rule**

Here we have three indices, wgtidxno1, wgtidxno2, and wgtidxno3. We can decompose the testing into three separate 2-dim tables which we analyze in a similar manner as the 2-dim tables. We would then apply a separate decreasing test to the 3-dim portion.

Create three 2-dim tables by selecting the 2-dimension sub-table elements (0, j, k), (i, 0, k), and (i, j, 0) for i,j,k=0,... For each 2-dimension sub-table, find the maximum value where the column has a non-zero weight.

1. This should be the same for all column values of all rows. Call this max2.
   - For each 2-dimension sub-table, the weights for row=0 and column=0,..., max2 are analyzed as a 1 dimensional weight table, so the weight for row=column=0 should be zero. (Actually, if the value at [0][0][0] is not equal to 0 then there should be an error.)

2. Calculate the maximum and minimum weights for this row excluding column=0. The maximum should be greater than zero and the minimum should be less than zero.

3. The weights should be monotonically decreasing with increasing column index.
   - For each 2-dimension sub-table, the weights for column=0 and row=0,..., maxrows are analyzed as a 1 dimensional weight table, so the weight for row=column=0 should be zero.

4. Calculate the maximum and minimum weights for this row excluding row=0. The maximum should be greater than zero and the minimum should be less than zero.

5. The weights should be monotonically decreasing with increasing row index.

   **Note:** For the 3 2-dim sub-tables, there are overlaps. So, in final analysis, you only need to check min, max and monotonic decr only in 3 cases for the whole 3dim. They are:
   - [0,j,0], [i,0,0] & [0,0,k]

6. Make sure that the rest of the table is decreasing:
   - wtgtval(i, j, k) >= wtgtval(i+1,j,k),
   - wtgtval(i, j, k) >= wtgtval(i,j+1,k), and
   - wtgtval(i, j, k) >= wtgtval(i,j,k+1)
Level 2 rules

A failure of any of these rules will generate a warning.

DOB Nval rule

This rule is triggered for Birth Date weights, i.e., when the cmpfunccode is "DATE" or "VR1D1C_DATE," the cmpspeccode is "DOB," and the wgttype is "NVAL."

For each wgtcode:
1. The disagreement weight is either 0 or non-existent.
2. The default agreement weight (numval=-1) is between 3.0 and 5.6.
3. All remaining weights are greater than 0.5.

AXP Sval rule

This rule is triggered for AXP (Address+Phone) weights, i.e., when the cmpfunccode is "AXP" and the wgttype is "SVAL." For each wgtcode:

The following tests apply for wgtcode=XACT:
1. The exact match weights should be between 200 and 1500.

The following tests apply for wgtcode=PARM:
1. INITIAL_ADJWGT should be between 500 and 800.
2. PHONETIC_ADJWGT should be between 400 and 700.
3. NICKNAME_ADJWGT should be between 400 and 700.
4. NICKMETA_ADJWGT should be between 500 and 800.
5. NICKMETA_ADJWGT should be more than PHONETIC_ADJWGT.
6. NICKMETA_ADJWGT should be more than NICKNAME_ADJWGT.
7. EDITDIST_ADJWGT should be between 400 and 700.
8. CELLDIFF_ADJWGT_2 should be between 200 and 400.
9. CELLDIFF_ADJWGT_3 should be less than 400 and more than CELLDIFF_ADJWGT_2.
10. CELLDIFF_ADJWGT_4 should be less than 400 and more than CELLDIFF_ADJWGT_3.

CXNM Sval rule

This rule is triggered for CXNM weights, i.e., when the cmpfunccode is "CXNM" and the wgttype is "SVAL." For each wgtcode:

The following tests apply for wgtcode=XACT:
1. The exact match weights should be between 300 and 800.

The following tests apply for wgtcode=PARM:
1. INITIAL_ADJWGT should be between 250 and 400.
2. PHONETIC_ADJWGT should be between 200 and 400.
3. NICKNAME_ADJWGT should be between 200 and 400.
4. NICKMETA_ADJWGT should be between 250 and 450.
5. NICKMETA_ADJWGT should be larger than PHONETIC_ADJWGT.
6. NICKMETA_ADJWGT should be larger than NICKNAME_ADJWGT.
7. EDITDIST_ADJWGT should be between 200 and 400.
8. CELLDIFF_ADJWGT_2 should be between 100 and 250.
9. CELLDIFF_ADJWGT_3 should be less than 400 and more than CELLDIFF_ADJWGT_2.
10. CELLDIFF_ADJWGT_4 should be less than 400 and more than CELLDIFF_ADJWGT_3.
11. NORM_ADJWGT_15 should be between 350 and 550.
12. NORM_ADJWGT_15 should be the largest value.
13. NORM_ADJWGT_0 should be between -300 and -100.
14. NORM_ADJWGT_0 should be the smallest value.
15. The values should decrease from NORM_ADJWGT_15 to NORM_ADJWGT_0.

**GENDER/SEX Sval rule**

This rule is triggered for gender weights, i.e., when the cmpfunccode is "EQVD," the cmpspeccode is "GENDER" or "SEX," and the wgttype is "SVAL." For each wgtcode:
1. The disagreement weight (strval="d") is between -3.5 and -1.0 or non-existent.
2. The male agreement weight (strval="M") is between 0.1 and 0.5.
3. The female agreement weight (strval="F") is between 0.1 and 0.5.

**PXNM Sval rule**

This rule is triggered for PXNM weights, i.e., when the cmpfunccode is "PXNM" and the wgttype is "SVAL." For each wgtcode:

The following tests apply for wgtcode=XACT:
1. The default agreement weight ("a") is between 4.0 and 5.6.
2. The disagreement weight ("d") is between -3.5 and -1.5.
3. The remaining weights are greater than 0.0.

The following tests apply for wgtcode=META:
1. The default agreement weight ("a") is between 3.0 and 4.0.
2. The disagreement weight ("d") matches the disagreement weight in the exact match (XACT) table.
3. All remaining weights are greater than 0.0.

The following tests apply for wgtcode=INIT:
1. The default agreement ("a") weight is 0.0 or non-existent.
2. The disagreement weight ("d") matches the disagreement weight in the exact match (XACT) table.
3. The remaining weights are greater than 0 and less than 4.5.

**QXNM Sval rule**

This rule is triggered for QXNM weights, i.e., when the cmpfunccode is "QXNM" and the wgttype is "SVAL." For each wgtcode, the PARM values that are used in QXNM are the following:

**Adjustment weights:**
INITIAL_ADJWGT, PHONETIC_ADJWGT, NICKNAME_ADJWGT, NICKMETA_ADJWGT, EDITDIST_ADJWGT, EDITDIST_FACTOR

Corresponding Min and Max values:

INITIAL_MINWGT, INITIAL_MAXWGT, PHONETIC_MINWGT, PHONETIC_MAXWGT, NICKNAME_MINWGT, NICKNAME_MAXWGT, NICKMETA_MINWGT, NICKMETA_MAXWGT, EDITDIST_MINWGT, EDITDIST_MAXWGT

For each wgtcode in the sval table with cmpfunccode of QXNM:

The following tests apply for wgtcode=XACT:
1. In general, higher frequency names will get lower weights. For example, JOHN, JIM, etc. may occur a lot of times thus getting a lower weight when compared to a name like TABULLAH.
2. The default agreement weight ("a") should be between 3.0 and 5.0
3. The disagreement weight ("d") should be between -3.5 and -1.5
4. Exact match weights for names are generally between 2.0 and 5.0

The following tests apply for wgtcode=PARM:
1. Full Max weight (FULLNAME_MAXWGT) is generally between 5.0 - 10.0 (or about 2.5 times the agreement weight)
2. All adjustment weights (_ADJWGT) should be greater than 0
3. All adjustment weights should be between their corresponding MIN and MAX parameter values. For example, INITIAL_ADJWGT should be between INITIAL_MINWGT and INITIAL_MAXWGT
4. All Min and Max parameters (_MAXWGT, _MINWGT) should be greater than 0
5. All adjustment weights should be less than MAX weight (FULLNAME_MAXWGT)
6. NICKMETA_ADJWGT should be greater than both NICKNAME_ADJWGT and PHONETIC_ADJWGT
7. INITIAL_ADJWGT should have the highest adjustment weight

AXP 1Dim rule

This rule is triggered for Address/Phone weights, i.e., when the cmpfunccode is "AXP" and the wgttype is "1DIM." For each wgtcode:
1. The weight value for index 1 should be between 400 and 800.
2. The weight value for index 8 should be between 900 and 1200.
3. The weight values should increase as the indexes increase.

DOB 1Dim rule

This rule is triggered for Birth Date/DOB weights, i.e., when the cmpfunccode is "DATE" or "VR1D1C_DATE," the cmpspeccode is "DOB," and the wgttype is "1DIM." For each wgtcode:
1. The second row (index 1) weight is 0.0.

Note: (This refers to an exact match on both date and month, so the weight is looked up in the mpi_wgtsval for year only.)
2. The third row (index 2) weight is between 1.5 and -2.5.
3. The fourth row (index 3) weight is between 0.5 and -3.5.
4. The fifth row (index 4) weight is between 1.0 and -4.5.

**SSN 1Dim rule**

This rule is triggered for Social Security weights, i.e., when the cmpfunccode is "DR1D1A," "DR1D1B," or "DR1D1C," the cmpspeccode is "SSN," and the wgttype is "1DIM." For each wgtcode:
1. The second row (index 1) weight is between 4.0 and 6.0.
2. The smallest value is between -5.5 and -2.0.

**AXP 2Dim rule**

This rule is triggered for Address/Phone weights, i.e., when the cmpfunccode is "AXP" and the wgttype is "2DIM." For each wgtcode:
1. The weight for position (0,1) which is the exact match on Phone and missing Address should be between 3.5 and 4.5.
2. The weight for position (1,0) which is the exact match on Address and missing Phone should be between 3.0 and 4.5.
3. The weight in position (1,1), which is the exact match on Address and Phone, should have the highest value.
4. The weight in position (1,1) should be between 4.5 and 6.0.
5. The lowest weight should be between -1.0 and -3.0.

**ZAP 3Dim rule**

This rule is triggered for ZIP/Address/Phone weights, i.e., when the cmpfunccode is "DR3D1A," "DR3D1B," or "DR3D1C," the cmpspeccode is "ZAP," and the wgttype is "3DIM." For each wgtcode:
1. The weight in position (1,1,1), which the exact match on Zip, Address and Phone, should have the highest value.
2. The weight in position (1,1,1) should be between 3.5 and 5.5.
3. Zip should have a maximum index of 5.
4. Address should have a maximum index of 15.
5. Phone should have a maximum index of 7.
Chapter 7. Analytics

IBM Initiate Workbench 9.7 provides a set of analysis tools for analyzing various aspects of the configuration, such as buckets and entities. The purpose of these tools is to evaluate the configuration and assist in finding errors and potential performance problems associated with the configuration. These tools are integrated with each other as well as with the other IBM Initiate Workbench configuration tools to assist in seamlessly configuring a Hub and validating the correctness of the configuration.

The Analytics view is first associated with a Hub from which it will extract data. Then, you select the queries to run within the view, each of which displays a specialized set of data. Available queries are categorized into Data Analysis, Entity Analysis, Bucket Analysis and Linkage Analysis types.

Table 61. Available Queries List

<table>
<thead>
<tr>
<th>Entity Analysis Queries</th>
<th>Bucket Analysis Queries</th>
<th>Data Analysis Queries</th>
<th>Linkage Analysis Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entities by Size Overview</td>
<td>Bucket Analysis Overview</td>
<td>Attribute Completeness</td>
<td>Member Overlaps</td>
</tr>
<tr>
<td>Entity Composition Bucket Size Distribution</td>
<td>Comparison Data Completeness</td>
<td>Member Duplicates</td>
<td></td>
</tr>
<tr>
<td>Entity Size Distribution Bucket Composition</td>
<td>Bucket Composition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Member Comparisons Bulk Cross Match Comparison Distribution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Member Entity Frequency Members by Bucket Count</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Member Entity Values Member Bucket Values</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Members by Entity Count Member Bucket Frequency</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Score Distribution Member Comparison Distribution</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Analytics View

About this task

The Analytics view provides query tools to the configuration user to analyze a hub configuration.
Accessing the analytics view

About this task

You can show the Analytics view from within the Configuration perspective by selecting **Window > Show View... > Workbench > Analytics** from the main menu. You can also open the Analytics perspective to show multiple Analysis views simultaneously. Four Analytics views appear by default within the Analytics perspective.

The icons at the top of the Analytics view provide the tools for accessing analytics data.

*Table 62. Analytics view toolbar icons*

<table>
<thead>
<tr>
<th>Icon</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Icon]</td>
<td>Set Analysis Source</td>
<td>To connect to the Hub from which analytics data is drawn. (Data is taken from the Hub’s database directly.)</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Previous query</td>
<td>To navigate to the previous query in the view. Use the small down arrow to select a query by name.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Next query</td>
<td>To navigate to the next query in the view. Use the small down arrow to select a query by name.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Add query</td>
<td>To add a query to the view.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Clear queries</td>
<td>To remove all queries from the view.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Export query</td>
<td>To save the currently displayed query to a CSV file.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Add new view</td>
<td>To create a new empty view within the perspective.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Pin query</td>
<td>To “pin” the query results to the current view and prevent drilldowns from changing the contents of the view.</td>
</tr>
</tbody>
</table>

Connecting to a Hub

About this task

The Analytics view must first be associated with a Hub to provide data for it. To associate a Hub with the view:

**Procedure**

1. Click on the **Hub database connection** icon in the Analytics view toolbar. The Analysis Source Configuration dialog opens.
2. Select Hub as the **Analysis source**.
3. Select the Hub, Member type and Entity type to which to connect.
4. If you wish to save the data to view later (as a Snapshot), enable the **Save Analysis Data to a Snapshot** option and provide a name in the **Analysis ID** field. Snapshots are saved in XML format to the “snapshots” folder in the Navigator view. By saving data into snapshots, you can compare analysis data from before or after a configuration change is made or from different points in time. Multiple copies of the same query can be saved within a single snapshot.
provided their input parameters are different. See “Loading previous analysis results” on page 190 for more information.

**Note:** All of the analysis views pull data directly from the database. Analyzing data not loaded into the database is not supported.

5. Click **OK**.
6. When prompted, provide a valid login name and password to access the Hub, then click **OK**. The Analytics view tab now includes the name of the Hub from which data is extracted for each analytics report.

### Adding a query to a view

**About this task**

Once the Analytics view has a data source associated with it, you can load one or more queries and view the results. To load a query:

**Procedure**

1. Click the **Add query** icon in the view toolbar. The Available Queries dialog opens.
2. Select the desired query and click **OK**. The query loads into the view.
3. Select any desired options or values, and then click **Run Query**. The view is refreshed with the appropriate data.
4. Multiple queries can be opened in each Analytics view. Use the **Next query** and **Previous query** toolbar icons to navigate between open queries. If desired, you can open multiple Analytics view instances and populate each with the desired set of queries. Open additional view instances by clicking on the **Add new view** icon in the view’s toolbar.
5. Some queries contain fields into which you can drill down to view more detailed data. If you want to “freeze” a view temporarily and force drilldowns to a different view, click the **Pin query** icon.
6. To remove all queries from a view, click the **Clear queries** icon.

### Exporting query results to CSV format

**About this task**

After being run, most queries can be exported to Comma Separated Values (CSV) format. To do so:

**Procedure**

1. Click the **Export query** icon on the view’s toolbar. Note that this icon will not be enabled until a query has been run. The Analysis Export dialog opens.
2. On the Analysis Export dialog, provide the desired values for the export options:
   - Export file - use the **Browse** button to set the path and file name
   - Field separator - a comma is the default
   - Generate header row within the file
   - Generate comments (query name and query parameters) within the file
3. Click **Finish**.
Loading previous analysis results

About this task

Analysis results can be saved into what is called a “snapshot.” When initially connecting to a Hub, just select the option to Save Analysis Data to a Snapshot and provide a snapshot name in the Analysis ID field.

To load a previously saved snapshot:

Procedure

1. Use an existing or new Analytics View. (To keep an existing Analytics View connected to a hub or another snapshot, create a new view by clicking the Add new view icon on the view’s toolbar.) You can also navigate to the Analysis perspective to view multiple Analysis views simultaneously.
2. Click the Set Analysis Source icon on the desired view.
3. On the Analysis Source Configuration dialog, set the Analysis Source to Snapshot. Note that drilling down into data is not available for snapshots.
4. Select the desired snapshot from the list of Available Snapshots.
5. Select the appropriate Member type and Entity type, and then click OK.
6. Click the Add query icon to select a previously saved query to view. The Available Queries dialog opens.

   Note: Multiple entity type and member type data can be saved to the same snapshot, but can only be viewed individually as a separate source.

7. Select the desired query to view. This list is typically shorter than that from a Hub analysis source. If multiple copies of the same query were saved in the snapshot, you can view their original input parameters in the Query Parameters table on the right.
8. Click OK.

Results

Multiple queries can be opened in the Analytics view. Use the Next query and Previous query toolbar icons to navigate between open queries.

Note: All queries selected from the snapshot are shown in a “read-only” mode where rerunning the query or performing drilldown queries are disabled.

Obtaining data

About this task

Each query contains a Run Query button that is used to execute the query. Some queries have optional input fields that are used as query criteria.

Depending on the particular query and the dataset size on the Hub, some queries may not return immediately. Queries that do not return immediately will run in the background and will populate the view when finished.

Note: If IBM Initiate Workbench is closed when the query finishes, you can still view the results of the query. To do this, open the Jobs View and navigate to the job that corresponds to the view’s query. Select the job, right-click and select Get Job Results. This will fetch the result from the Hub and then populate the view.
The query results on the server are not persisted and will be lost if the server is restarted. The results are not saved locally into a snapshot until they are displayed in an Analytics view (either synchronously for short running queries or after performing the “Get Job Results” action for longer running queries).

**Preparing analysis data for large datasets**

**About this task**

If the number of records in the Hub is larger than 2 million, the bucket analysis queries will not execute unless the data is first “prepared.” Data preparation involves taking the raw member and bucket data and pre-computing an intermediary set of data that can be quickly queried. This data preparation can be done through the “Bucket Analysis Preparation” job. Preparing data for 2-5 million records should take around 10 minutes, while preparing data for 50 million records will take around 5 hours. These estimates may vary wildly depending on different hardware and database configurations. If the member data is modified, then the prepared data should be recomputed as well to avoid seeing out-of-date results.

**Note:** The two million-record threshold is configurable and can be changed through JMX.

### Entity Analysis

The entity analysis tools are packaged within IBM Initiate Workbench in a pre-defined set of queries called the “Entity Analysis” queries.

Any of these queries can be added to the Analytics view along with other analysis tools (Bucket analysis, etc.).

#### Entities By Size

This query provides the ability to query for entities that match a specified range of sizes (number of members in an entity). Specifying a value of 0 for either the minimum or maximum size indicates that there is no limit (no minimum or no maximum).

The table shows the member count and the entity id.

Individual entities can be analyzed by selecting a row in the table and clicking on the View Entity button. This will run and display the Entity Composition query on the selected entity.

#### Entity Composition

This query shows the content of a specified entity. The table lists the member record IDs and source IDs that are in the specified entity as well as the comparison data for each member. The comparison data is split out by comparison role into individual columns of the table.

Selecting a row in the table and clicking on the Compare Members button will bring up the Member Comparisons view and run a query to compare the selected member against all other members in the selected entity (including comparing the selected member to itself). The selected member can also be compared to a subset of other members in the entity by control-clicking on additional members in the table and then clicking the Compare Members button.
**Entity Size Distribution**

This query provides a comprehensive view of all the entities in the Hub as they relate to size. The view may be filtered to show entities from the checked sources only. If an entity is comprised of members in a checked source(s) as well as an unchecked source(s), then the size shown for the entity will be a count of the member records in the checked sources only.

Large/suspect entities can be further analyzed by clicking on the desired bar in the chart. This will bring up the Entities By Size view and run a query to show entities of that size. By pressing the Ctrl key before clicking on the bar the query will show entities of that size and larger.

**Note:** Values on the x-axis that do not show a bar have at least 1 entity matching the size specified on the x-axis but not enough members to make the bar visible in the chart.

**Member Comparisons**

This query provides a mechanism to compare a member record against all the members in a specified entity or to a set of specified members. To compare against multiple individual members just click the **Specified Member(s)** button and type in the member record ID(s) separated by a comma.

The results are displayed in a tree that can be expanded and/or collapsed to suit the data visibility needs. For each row in the table corresponding to a member, a score will be provided. By expanding the member, the individual comparison elements can be inspected and the individual weights for each comparison specification can be analyzed. Both the proband and candidate member comparison data is displayed along with the match type used to come up with the individual weight.

**Note:** The use of the term “weight” here is historical and should not be confused with “weights” or “generated weights.” In this instance a weight is simply a component of a score. The score is the aggregated total of all the weights for that comparison.

Possible match types, based on the attribute comparison score, are:
- **E** = equal (the values are the same)
- **D** = different (the values are different)
- **P** = partial (values are partially the same)
- **M** = missing (value is missing)
- **X** = EQWRD (equal at a ‘word’ or token level)
- **Y** = EQINI (Equal at a single character initial level)
- **P** = phonetic (values have a phonetic match)
- **N** = nickname (nickname value)
- **O** = NICKMETA
- **I** = initial (Equal at an initial level, but not a direct match of a single character.
- **A** = acronym
- **C** = compound match
- **1** = Prefix match with prefix in 1st member
- **2** = Prefix match with prefix in 2nd member
You may also see a value of T (true) or F (false). This occurs if the False Positive Filter (FPF) feature is used:

- T indicates that the FPF triggered, and the algorithm believes the records are not the same person.
- F indicates that the FPF was not triggered, and the records may be the same person.

The associated score is dependent on the settings in the weight table.

Selecting a row in the table and clicking the View Algorithm button will open the algorithm for that member's member type. If a row with an individual weight is selected, then the comparison function that was responsible for that weight will be selected.

The table can be sorted by the candidate member ID or by score.

**Member Entity Frequency**

This query shows the frequency in which members appear in entities; that is, the number of members who are in one entity, the number who are in two entities, the number who are in three entities, and so on.

**Member Entity Values**

This query shows the entities to which a member belongs.

**Members by Entity Count**

This query shows a list of members who are in a specified range of entities (for example, all members who are in 3 or more entities). A value of 0 in the Maximum # of entities field means no maximum number is specified, otherwise the Maximum # of entities value must be greater than or equal to that in Minimum # of entities.

**Score Distribution**

This query shows the distribution of scores for all the record pairs in the system. Single member entities or entities with more than 2 member records are not included in the results.

The number of pairs for each score is actually the sum of all counts in a given score range. For example, an x-axis score value of 27 represents all pairs that score between 26.1 and 27.0.

You can use the Source Filter tab to filter the view to show entities from the checked sources only. If an entity is comprised of members in a checked source(s) as well as an unchecked source(s), then the size shown for the entity will be a count of the member records in the checked sources only.

If no results show for a particular linkage type, there may not be any entities meeting the criteria for that linkage type and/or set of selected sources.

**Note:** Values on the x-axis that do not show a bar have at least 1 entity matching the size specified on the x-axis but not enough members to make the bar visible in the chart.
Data Analysis

Currently, two data analysis tools are packaged within IBM Initiate Workbench and listed in the Available Queries dialog in the “Data Analysis” section.

These queries can be added to the Analytics view along with other analysis tools (Entity analysis, etc.).

Attribute Completeness

Formerly called the Attribute Validity report, this query shows the percentage of time the records from selected sources have values for the member types attributes. Values that are present in high percentages should be considered as potential candidates for use in algorithms.

By default, the results are sorted by attribute name. The results can also be sorted by any of the columns by clicking on the column header.

Also noted in the table is the percent of the member type's records that are contained in a specified source. This information is shown in the column header next to the source name.

Comparison Data Completeness

This report calculates the percentages of comparison roles by source that have at least one and more than one occurrence of member data. Values that are present in high percentages should be considered as potential candidates for use in algorithms.

This report uses the comparison data for each member, meaning that the anonymous values are filtered out. The system offers you the option of removing anonymous values from the Attribute Completeness report. Implementers would get a more accurate picture of attribute completeness by removing anonymous values such as BABYBOY, that are essentially treated as nulls, from the Attribute Completeness report.

By default, the results are sorted by comparison roles. The results can also be sorted by any of the columns by clicking on the column header.

Also noted in the table is the percent of the member type's records that are contained in a specified source. This information is shown in the column header next to the source name.

Bucket Analysis

Several bucket analysis tools are packaged within IBM Initiate Workbench and listed in the Available Queries dialog in the “Bucket Analysis” section.

Any of these queries can be added to the Analytics view along with other analysis tools (Entity analysis, etc.).

Bucket Analysis Overview

This query provides some general information on the health of the Hub's bucketing strategy.
The top half of the view is filled with information such as number of large buckets, unbucketed members, etc. To view a particular range of large buckets and/or unbucketed members click the appropriate button. Clicking on a View Buckets button will select the Buckets By Size view and run a query with the desired range of bucket sizes. Clicking on the View Members button will select the Members By Bucket Count view and run a query to show members without any buckets.

The Largest 10 Buckets tab shows the ten largest buckets along with those buckets' hash values, the bucket role that generated the bucket, as well as a bucket value from one of the members in those buckets (which will typically be identical for all members in the same bucket). Selecting a bucket hash and clicking on the View Bucket button will run the Bucket Composition query and populate the view with the select bucket's members and those member's bucket values for that hash code.

**Bucket Composition**

This query shows the content of a specified bucket. The table lists the memrecnos that are in the specified bucket as well as the bucket role and bucket value for each member in that bucket. The bucket values shown are the actual bucket values freshly calculated from the member data in the database. If different bucket values show up for the same bucket hash then that would indicate a bucket hash collision. This would be considered an anomaly and might explain why certain members are being compared against each other which normally would not compare against each other. However, such a condition is not in general considered hazardous to the system's health.

Selecting a row in the table and clicking the View Member button will run the Member Bucket Values query to show all of the selected member's buckets. Clicking the View Algorithm button will open the algorithm editor and select the bucket role that created the specified bucket.

**Bucket Size Distribution**

This query provides a comprehensive view of all the buckets in the Hub as they relate to size. Large buckets are shown to the right side of the view and are indicated by a color indicator that goes from green (smaller buckets) to yellow (medium sized buckets) to red (large buckets). The data points in the graph will typically follow a downward curve from the left to the right. Extensive data points on the right side (in the red area and/or outside of the normalized slope of the curve) are typical areas of concern and could indicate missed anonymous values, incorrect thresholds, data problems, etc.

Clicking on a data point will select the Buckets By Size view and will run a query to show those buckets of that size. By pressing the control key before clicking on the data point and query will show those buckets of that size and larger.

**Buckets By Size**

This query provides the ability to query for buckets that match a specified range of sizes (number of members in a bucket). Specifying a value of 0 for either the minimum or maximum size indicates that there is no limit (no minimum or no maximum).

The table shows the member count, the bucket hash, bucket role, and a sample bucket value from one of the members in the bucket. Typically the bucket value will be the same for all members in any given bucket. The exception to this is if there was a hash collision that resulted in different bucket values having the same
bucket hash. To check this condition, you can select the bucket and click the View Bucket button to view all of the members and their bucket values for any given bucket.

The table can either be sorted by the number of members in the bucket or by the bucket role. To change the sorting, click the appropriate table header column.

If it is determined that a problem exists with a particular bucket role (lack of frequency based bucketing, etc.), the algorithm editor can be opened by selecting a table row and clicking the View Algorithm button. This will bring up the algorithm editor and select the particular bucket role that created the selected bucket.

**Bulk Cross Match Comparison Distribution**

This query calculates the number of comparisons required for a bulk cross match as it relates to the maximum bucket set size parameter (Bucket Size Limit) that is specified on an mpxcomp job.

This number of comparisons can then be used together with the number of threads and number of comparisons per thread per second to determine the approximate completion time for a bulk cross match.

**Member Bucket Frequency**

This view answers the question “How many members are in 1 bucket, 2 buckets, 3 buckets, etc.” An x-axis data point of 0 shows the number of un-bucketed members.

Clicking on a bar in the chart will select the Members By Bucket Count view and run a query to show those members with that many buckets.

**Note:** Values on the x-axis that do not show a bar have at least 1 member for that bucket count but not enough members to make the bar visible in the chart.

**Member Bucket Values**

This view shows what buckets a specified member is in. The result table shows the bucket hash, bucket value, and the bucket role that produced each bucket.

Selecting a bucket and clicking the View Bucket button selects the Bucket Composition view and runs a query to show the bucket composition for the selected bucket hash. Clicking on the View Algorithm button opens the Algorithm editor and selects the bucket role that was responsible for creating that bucket.

**Member Comparison Distribution**

This view shows estimated performance of the system as it relates to the number of comparisons being performed. That is to say: when a search is performed, how many actual comparisons will be made? In the case below, the chart is telling us on average that 3 comparisons will be made. 1 in 10 (the lowest red line) comparisons will result in approximately 6 comparison, 1 in 100 would be 7.5, and 1 in 1000 comparisons would result in about 8 comparisons.

The data is based on 20,000 randomly sampled members from the system. If there are less than 20,000 members in the system, all members are used. On average, a target member will be compared against all members that share buckets with that target member.
Members By Bucket Count

This view provides a query for members based upon the number of buckets a member is contained in. Specifying a minimum and maximum of 0 will return all unbucketed members. For a minimum of greater than 0, a maximum of 0 indicates no limit.

The result table shows the memrecno, the number of buckets the member is in, as well as the cmpd string for that member. Selecting a member and clicking the View Member button selects the Member Bucket Values view to show all buckets that the member appears in.

Linkage Analysis

Currently, two data analysis tools are packaged within IBM Initiate Workbench and listed in the Available Queries dialog in the “Linkage Analysis” section.

These queries can be added to the Analytics view along with other analysis tools (Entity analysis, etc.).

Member Duplicates

This query shows the various error rates around duplicate members (member records from the same source that link to the same entity).

The first four columns show the raw data from the Hub database (broken down by source): number of members, number of entities, number of duplicate sets, and the number of members in those duplicate sets. The last 3 columns are the various error rates that can be calculated from those values.

- Record Error Rate – Indicates how many records you have to look at to resolve your duplicates, or how many records have an incomplete view of a member.
- Entity Duplication Rate – Indicates how many members have duplicate records, or the probability that a random member has a duplicate record.
- Record Duplication Rate – Indicates how many records are duplicates, or perhaps the percentage of records that could be eliminated.

Member Overlaps

This query provides information on the number of overlaps in the hub.

An overlap exists when an entity has records from multiple sources. For example, if an entity with three records exists, and each record is in a separate source system, then each source would be said to have two overlaps in it (A with B, A with C, et cetera).

The first column group just shows the number of unique entities represented in the specified source as well as the percentage of all entities that are represented by a record in that source.

The second column group shows the count and percent of those entities that have overlaps in at least one other source (those entities have at least one record in another source). Entities with overlaps in multiple other sources are only counted once in these two columns.

The remaining column groups are for each source by source combination. When the row and column source is the same, the count is simply the count of entities in that source, and percent will always be 100%. However, when the row and column
sources are unique, the count represents the number of overlaps that exist between
the row source system and the column source system. The percent value then
represents the percent of entities in the row source that have overlaps in the
column source.
Chapter 8. Callout handlers

You may want to send notifications or API calls to external systems when certain events have occurred within the Hub, such as entity linking/unlinking, member data modifications, member searches and so on.

This feature enables you to register custom callout handlers that manage your special business needs, such as:

- Callouts to a third-party API
- Apply conditional security for certain types of interactions
- Send event notifications to an external message queue, e-mail address, file system, and so forth.

All of these tasks are accomplished by registering and deploying custom java handlers, leveraging the classes found in madapi.jar and madhandlers.jar.

Each handler is registered and deployed on a specific IBM Initiate Master Data Service software instance. At the time of registration, you specify the interaction types that will trigger it, as well as the callback type. The callback type determines when, in relation to the interaction, the handler is invoked.

Note: When creating handlers for use with Clover.ETL graphs, refer to “Special considerations for Clover.ETL graphs” on page 203 for packaging and deployment instructions.

Custom handlers

Your custom handler will be executed in anticipation of or in response to an interaction occurring within the Hub or an entity manager. What the handler does, specifically, is up to you. Examples include:

- Sending a brief message to a pager or mail address
- Posting a message in a messaging queue
- Calling a third-party rules Hub API
- Applying special security to allow or disallow the interaction
- Updating data in the hub database

Custom handlers are created using one of the Initiate SDKs. Consult the IBM Initiate Master Data Service SDK Reference for Java and Web Services for more information on creating custom handlers. Once handlers are created and packaged, they are registered and deployed using IBM Initiate Workbench.

Enabling engine callouts

About this task

Setting the MAD_CALLBACKLIB environment variable enables the Hub to call custom handlers. Use one of the following values for the MAD_CALLBACKLIB environment variable:

- mpicbjava.dll – Java-wrapped engine only, Microsoft Windows environments
libMPICBJAVA.so – Java-wrapped engine only, environments such as IBM AIX®,
Linux, Solaris, or HP-UX.
mpicbdotnet.dll – Java-wrapped or non Java-wrapped engine, Microsoft
Windows only

To set MAD_CALLBACKLIB:

**Procedure**
1. Open the `/inst\mpinet_hub_instance_name\conf\`
   `com.initiate.server.system.cfg` file in a plain text editor.
2. Supply the appropriate value for
   MAD_CALLBACKLIB=
3. Restart the Hub after modifying this value.

---

### Packaging the handler

When you create an Initiate project, the New Initiate Project wizard creates a
handlers directory containing two Ant files: `package-handlers.xml` and
`package-handlers.properties`.

**About this task**

These files support the packaging of Java handlers developed within a project. The
source for Java handlers must reside in the project `src` directory, which allows the
classes to be incrementally compiled by IBM Initiate Workbench. The
`package-handlers.xml` script assumes that the compiled Java handler classes reside
in the project `bin` directory, which is the default location for compiled Java classes
in an Initiate project. The `package-handlers.properties` file is optional, but can be
used to override any default property values defined in the `package-handlers.xml`
script, such as the location of the compiled Java classes, which Java classes to
include or exclude in the package, and so on.

**Attention:** Deployment of signed engine callout JARs is not supported. If signed
third-party JAR files are in the `/lib` directory, you must manually remove the
signatures before you deploy the callout. If the signatures are not removed, an
exception is issued during deployment.

### Adding external libraries

Any libraries required by the Java handler classes should be added to the project
`lib` directory. In order to compile the Java handler classes that depend on these
third-party libraries, you must add the libraries to the project build path.

**About this task**

Before adding libraries, be aware that deployment of signed engine callout JARs is
not supported. If signed third-party JAR files are in the `/lib` directory, you must
manually remove the signatures before you deploy the callout. If the signatures are
not removed, an exception is issued during deployment.

**Procedure**
1. After the library JAR file is copied to the project `lib` directory and is visible in
   the Navigator, select **Project > Properties** from the main menu.
2. On the Properties for `ProjectName` dialog, select **Java Build Path** on the left
   pane and the **Libraries** tab on the right.
3. Click the Add JARs... button.

4. On the JAR Selection dialog, select the JAR file from the `project_name`\>lib path, and then click OK.

5. On the Properties for ProjectName dialog, click OK.

### Creating the launch configuration

**About this task**

To execute the package-handlers.xml Ant script within IBM Initiate Workbench, you must first create a launch configuration for it:

**Procedure**

1. Expand the handlers directory and right-click on the `package-handlers.xml` script.
2. Select Run As from the context menu, then select Ant Build.... The Edit configuration and launch dialog opens.
3. On the Refresh tab, check the Refresh resources upon completion option, and then select The project containing the selected resource option.
4. Click Apply to save the changes.
5. Click Run to launch the Ant script.
6. As the package-handlers.xml script executes, progress messages are displayed in the Console view.

**Results**

If the script executes successfully, the handlers directory will contain a `com.initiate.server.handler.ext.jar` file containing the project's Java handler classes. This Java handlers package can be deployed to a Hub using the “Deploy Handlers” wizard. See “Deploying the handler” on page 205.

### Re-running the Ant script

**About this task**

After the initial launch configuration is created and executed for the Ant script, it can be re-run by clicking the Run MDE Project package-handlers.xml shortcut icon on the toolbar:

### Registering the handler

**About this task**

Handlers are registered in IBM Initiate Workbench in the Handlers view. See “Handler metadata” on page 202 for descriptions of the metadata. Refer to “Available interaction codes” on page 203 for a list of interaction codes.

The registration process varies slightly depending on the environment/language used.
Java handlers

About this task

The steps for registering and deploying handlers when using the mpicbjava.dll or libMPICBJAVA.so libraries are:

Procedure

1. Add, edit or delete handler registrations in IBM Initiate Workbench. Refer to the detailed instructions below.
2. Deploy the configuration to the Hub.
3. Deploy the handler to the Hub.

.NET handlers

About this task

The steps for registering and deploying handlers when using the mpicbdotnet.dll library are:

Procedure

1. Deploy the handler to the Hub.
2. Add, edit or delete handler registrations in IBM Initiate Workbench. Refer to the detailed instructions below.
3. Deploy the configuration to the Hub.

Handler metadata

Handlers are registered in the Handlers view. Each handler registration consists of the following metadata:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callback type</td>
<td>The point at which the handler can be called: ALL – used for handler classes which contain preIxn(), postIxn() and/or entMng() methods. preIxn - executed immediately preceding the action specified by the interaction code. postIxn - executed immediately after the action specified by the interaction code. entMng - executed immediately after the entity manager has run.</td>
</tr>
<tr>
<td>Interaction code</td>
<td>The interaction to trigger a call to the handler. The available Interaction codes are listed in the table below.</td>
</tr>
<tr>
<td>Sequence number</td>
<td>The order the handler is applied. The Sequence number must be unique for any given Callback type/Interaction code combination. This enables multiple handlers to be called in a predetermined sequence. (You can also register the same handler class multiple times by setting a different Sequence number for each.)</td>
</tr>
<tr>
<td>Handler Class</td>
<td>Handler implementation: Java - &lt;full_package&gt; (e.g. com.initiatesystems.handler.FileHandler) .NET - &lt;dll_filename&gt;:&lt;package_namespace&gt;:&lt;handler_classname&gt; (e.g. FileHandler.dll:handler.FileHandler)</td>
</tr>
</tbody>
</table>
### Property Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handler arguments</td>
<td>Additional arguments passed to the handler init() method. Separate arguments with the carat (^) character.</td>
</tr>
</tbody>
</table>

### Available interaction codes

These interaction codes are found in the ixncode column of the mpi_ixnhead table, with the exception of ALL. Descriptions of interactions can be found in the IBM Initiate Master Data Service SDK Reference for Java and Web Services.

**Table 63. Available interaction codes**

<table>
<thead>
<tr>
<th>MEMGET</th>
<th>MEMCOMPUTE</th>
<th>EIAPUT</th>
<th>USRGETINFO</th>
<th>ENGSETINFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEMPUT</td>
<td>MEMGENKEY</td>
<td>EIASEARCH</td>
<td>APPGETINFO</td>
<td>ENGGETINFO</td>
</tr>
<tr>
<td>MEMUNPUT</td>
<td>MEMDELETE</td>
<td>TSKGET</td>
<td>NETGETINFO</td>
<td>ENGSQLEXEC</td>
</tr>
<tr>
<td>MEMDROP</td>
<td>MEMUNDELETE</td>
<td>TSKPUT</td>
<td>USRSEARCH</td>
<td>MPXINIT</td>
</tr>
<tr>
<td>MEMMATCH</td>
<td>MEMCREATE</td>
<td>TSKSEARCH</td>
<td>GRPGETINFO</td>
<td>MPIRETSK</td>
</tr>
<tr>
<td>MEMSEARCH</td>
<td>MEMPOSTIT</td>
<td>DICGET</td>
<td>GRPSEARCH</td>
<td>MPILOADI</td>
</tr>
<tr>
<td>MEMCOMPARE</td>
<td>MEMPUTBULK</td>
<td>DICPUT</td>
<td>AUDGET</td>
<td>MPILOADU</td>
</tr>
<tr>
<td>MEMMERGE</td>
<td>MEMUNDO</td>
<td>DICGENKEY</td>
<td>AUDPUT</td>
<td>ENTMNGMEM</td>
</tr>
<tr>
<td>MEMUNMERGE</td>
<td>EIAGET</td>
<td>USRSETPASS</td>
<td>AUDSEARCH</td>
<td>MPIRETSK</td>
</tr>
<tr>
<td>CRAPPLY</td>
<td>CRCONFIGGET</td>
<td>CRCONFIGPUT</td>
<td>CRDELETE</td>
<td>CRDETAIL</td>
</tr>
<tr>
<td>CRGET</td>
<td>CRSEARCH</td>
<td>CRUPDATE</td>
<td>WDEMOTE</td>
<td>WFPROMOTE</td>
</tr>
</tbody>
</table>
| MEMTEXTSEARCH | NEMTERMSearch | QUEUEGET | QUEUEPUT    | QUEURESET   
| ALL         |             |         |             |             |

### Special considerations for Clover.ETL graphs

#### About this task

When building handlers for use with Clover.ETL, the following basic steps should be followed:

**Procedure**

1. Enable the engine callout. Set the MAD_CALLBACKLIB to mpi_cbjava.dll. See "Enabling engine callouts" on page 199.
2. Add the handler using Clover.ETL metadata. Refer to Table 64 on page 204.
3. Deploy the hub configuration.
4. Copy the Clover.ETL graph to the Workbench project's src directory.
5. Create the launch configuration. Refer to "Creating the launch configuration" on page 201.

**Note:** When you create the launch configuration, the graph is automatically packaged with the .jar file that is created, which is deployed in the next step. Therefore, you do not need to deploy the graph itself as a separate step.

6. Deploy the handler. Refer to "Deploying the handler" on page 205.
Clover handler metadata

When packaging Clover handlers, use the following metadata. Where Clover-specific metadata is not given, use standard handler metadata as described in “Registering the handler” on page 201.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HandlerClass</td>
<td>For Clover graphs, use com.initiatesystems.hub.handler.CloverETLHandler</td>
</tr>
<tr>
<td>Handler arguments</td>
<td>Additional arguments passed to the handler init() method. Separate arguments with the carat (^) character.</td>
</tr>
<tr>
<td></td>
<td>The following arguments are specific to Clover graphs. The first two, graph and attributes are required, and if they are not present, the graph will not run.</td>
</tr>
<tr>
<td></td>
<td>• graph: the path to the embedded application. This argument is required for Clover graphs.</td>
</tr>
<tr>
<td></td>
<td>• attributes: a semicolon-separated list of all attribute names which are processed in the Clover graph.</td>
</tr>
<tr>
<td></td>
<td>• inputDictionaryEntry: the name of a dictionary entry for the input member row list. The default is inputMember.</td>
</tr>
<tr>
<td></td>
<td>• outputDictionaryEntry: the name of a dictionary entry for the output member row list. The default is outputMember.</td>
</tr>
<tr>
<td></td>
<td>• plugins: the path to the Clover engine plugins. The default is /lib/engine/etlplugins.</td>
</tr>
<tr>
<td></td>
<td>• context: the name of a dictionary entry for the Initiate context (default is context).</td>
</tr>
<tr>
<td></td>
<td>• maxGraphs: the maximum number of embedded graph instances preallocated in the graph pool; this is also the maximum number of parallel-running graphs. The default is 0, which equals unlimited.</td>
</tr>
<tr>
<td></td>
<td>• maxNodes: the maximum number of parallel-running nodes within all graphs. The default is 0, which equals unlimited.</td>
</tr>
</tbody>
</table>

Adding a handler registration

Procedure
1. Click the Add button in the Handlers view. A new handler registration is added with default property values.
2. Set the appropriate values for each property directly in the Handlers table. Refer to “Registering the handler” on page 201 for property descriptions.
3. If desired, enter a description of the handler in the Properties view.
4. If more than one handler is registered, use the Move up or Move down buttons as needed to set the order in which the handlers are to be executed.
5. Save the project. Once the project is deployed, the handler will be registered.
Editing a handler registration

Procedure
1. Navigate to the Handlers view and make any necessary changes to the properties directly in the Handlers table.
2. If more than one handler is registered, use the Move up or Move down buttons as needed to set the order in which the handlers are to be executed.
3. Save the project. Once the project is deployed, the handler registration will be complete.

Deleting a handler registration

Procedure
1. Navigate to the Handlers view and select the handler to unregister in the Handlers table.
2. Click the Remove button to unregister a handler.
3. Save the project. Once the project is deployed, the handler will be unregistered.

Deploying the handler

Use this procedure to remove a handler from your Hub configuration.

About this task
Handlers are deployed in IBM Initiate Workbench by using the Deploy Handlers wizard.

Attention: Deployment of signed engine callout JARs is not supported. If signed third-party JAR files are in the \lib directory, you must manually remove the signatures before you deploy the callout. If the signatures are not removed, an exception is issued during deployment.

Procedure
1. From the Initiate menu, select Deploy Handlers... The Deploy Handlers wizard opens.
2. Select the Project in which the handler was registered and the Hub to which you want to deploy it.
3. Select whether the handler to be deployed is a Java or .NET handler.
4. Use the Browse button to indicate the location of the handler package JAR file. The handler package file should be named com.initiate.server.handler.ext.jar.
5. Click Finish.

Testing the handler

About this task
Deploy new handlers on a test Master Data Engine instance before you deploy them into the production instance.

Procedure
1. Enable debug- or trace-level logging in the hub instance path\inst\mpinet_hub_instance_name\conf\log4j.xml file. For instructions on setting log
levels, consult the “Configuring Environment Variables and Settings” section in the IBM Initiate Master Data Service Engine Installation Guide.

2. In the client application, perform an interaction that will trigger the handler. For example, use IBM Initiate Inspector to execute a Member Get or Search operation.

3. View the instance engine log file (hub_instance_path\log\mpinet_hub_instance_name*.mlg) to verify that the handler was executed.

4. The debug- and trace-level logging is extremely verbose, so revert the settings after you have made sure the handler runs as expected.
Chapter 9. User management

If the User and Group Management option was selected during IBM Initiate Workbench installation, you can access the User Management perspective and the LDAP perspective. Both perspectives enable you to add and maintain access for those individuals authorized to use Master Data Engine software and associated applications.

- The User Management perspective is a custom plugin which enables you to easily add users and assign them to groups without any LDAP knowledge. This perspective works only with the internal LDAP server and does not display the fully-qualified LDAP distinguished names (DNs).
- The LDAP perspective is an Apache Directory Studio plugin which gives users familiar with LDAP the most power and flexibility in creating groups and users, and assigning users to groups. This perspective works with internal or external LDAP servers.

To access any application, each user must have a user name and password defined in the LDAP server. Methods for defining user names and passwords are discussed in the sections below.

After you add users, you can control their access to attribute types, composite views, interactions and operations by setting permissions for the LDAP groups to which they belong (see "Groups" on page 145 for details on group configuration).

Users who need permission to log in to IBM Initiate Workbench and perform tasks such as retrieving or uploading a hub configuration, defining user group permissions or running jobs must belong to the Administrators group, discussed in "Internal LDAP server."

Note: Because Initiate manages users and groups jointly, all users and groups must be maintained together either on an internal LDAP server or on an external LDAP server. Initiate does not support internal users with external groups, or vice versa.

Internal LDAP server

During the initial Hub configuration, you installed either an embedded or standalone Initiate internal LDAP server. If your organization does not currently have an LDAP server in use, you will need to use the internal one. If your organization currently uses an existing LDAP server to manage users and groups, you can grant existing users access to Initiate applications by setting group permissions. See "Groups" on page 145 for details.

Of the various attributes that can be used to locate a user in the LDAP directory, for the internal Initiate-specific LDAP directories, the cn is used.

The internal LDAP server comes pre-configured with the following groups:
1. cn=Administrators,ou=System,ou=Groups,dc=initiatesystems,dc=com
2. cn=Default,ou=System,ou=Groups,dc=initiatesystems,dc=com
3. cn=All Application
   Operations,ou=System,ou=Groups,dc=initiatesystems,dc=com
4. cn=All Composite Views,ou=System,ou=Groups,dc=initiatesystems,dc=com
5. cn=All Interactions,ou=System,ou=Groups,dc=initiatesystems,dc=com
6. cn=All Segments Read Write,ou=System,ou=Groups,dc=initiatesystems,dc=com
7. cn=All Segments Read Only,ou=System,ou=Groups,dc=initiatesystems,dc=com

The Administrators group
(cn=Administrators,ou=System,ou=Groups,dc=Initiatesystems,dc=com) is required
in the embedded and standalone LDAP server. Even if you are using an external
LDAP directory for your user base, there will always be either an embedded or
standalone LDAP server to store the internal Initiate system users and groups.

The Administrators group is pre-configured to have full access to IBM Initiate
Workbench to import and deploy Hub configurations, run Analytics reports, set
user group permissions and execute jobs on the Hub. Any users you wish to have
this access must be added to the Administrators group. To be part of the
Administrators group, the user has to be present in the internal directory server.
(See “Importing or exporting LDAP groups and users” on page 212 for instructions
on importing users from an external LDAP server.)

**Note:** Do not delete the system user or remove the system member from the
Administrators group without first creating at least one other user with Administrators
group membership. Also, the Administrators group must not be renamed.

IBM Initiate Workbench comes with a preconfigured user, system, (the default
password is also system) which has membership in the Administrators group. You
will use this login initially to add additional users and groups. Change the system
password to secure the installation.

The Default group, listing 2 above, is assigned to a user when he logs in if he is
not currently a member of any other defined group. This group cannot be granted
permissions through the IBM Initiate Workbench configuration editor. Refer to the
following table for a list of pre-defined functionality for each of these default
groups.

**Note:** These groups are not available to configure in IBM Initiate Workbench; this
list is provided for information only. User groups should be created according to
the needs of your organization and assigned permissions as described in “Groups”
on page 145.

<table>
<thead>
<tr>
<th>Group</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrators</td>
<td>Has full access to all interactions, operations, composite views, and attributes (segments).</td>
</tr>
<tr>
<td>Default</td>
<td>Has access to interactions USRGETINFO, GRPGETINFO, USRSETPASS. Has read only access to segments USRHEAD, GRPHEAD, GRPXAPP, GRPXCVW, GRPXIXN, and GRPXSEG.</td>
</tr>
<tr>
<td>All Application</td>
<td>Has access to all operations (appperm=A)</td>
</tr>
<tr>
<td>Operations</td>
<td>Has access to all composite views (cvwperm=A)</td>
</tr>
<tr>
<td>All Interactions</td>
<td>Has access to all interactions (ixnperm=A)</td>
</tr>
<tr>
<td>All Segments Read Write</td>
<td>Has read/write access to all attributes (segperm=W)</td>
</tr>
<tr>
<td>All Segments Read Only</td>
<td>Has read-only access to all attributes (segperm=R)</td>
</tr>
</tbody>
</table>
If you want a certain user to always have read/write permissions to all attributes, for example, you can assign that user to the All Segments Read Write group. Doing so gives the user permissions to read and write all attributes regardless of any permissions that might also be granted him via the Groups tab. This is helpful for non-human users, such as those created to enable handlers, custom applications and so forth.

Navigating the User Management perspective

About this task

The User Management perspective contains two views that enable users without an LDAP background to create, edit and delete users, and assign them to groups. New groups cannot be created in this perspective; use the LDAP perspective to add or delete groups. (See “LDAP groups” on page 213 for more information.)

Open the perspective by clicking the User Management icon on the perspective shortcut bar, or by selecting Window > Open Perspective > Other... and selecting the User Management perspective.

The User Management views typically connect to the LDAP server associated with the project’s Hub, although any LDAP server can be specified.

Procedure

1. Click the Connect icon on the view’s toolbar.
2. On the Connection Information dialog, enter the connection information.
   - Host - the LDAP host name or IP address.
   - Port - the LDAP port. The default port is 1389.
   - SSL Enabled - indicates whether SSL will be used to communicate with the internal directory server.
   - User - a user name which is a member of the Administrator’s group.
   - Password - the password for the selected user.
3. Click OK. The Users and Groups views are populated with information from the LDAP server.

Adding users

Procedure

1. On the Users tab, click New.... The fields on the right side of the screen are cleared and required fields are highlighted.
2. Enter a unique Login Name and the desired Password, and confirm the password by typing it in the Confirm field. The user should be instructed to change his password when he initially logs into the application.
3. In the Group Membership box, select the groups to which this user will have membership. The Default group is selected by default. If desired, use the All button to select all groups, or the None button to deselect all groups.

   Note: Users are not required to have membership in any groups, however, without membership in the Default group (at a minimum), the user cannot log into Initiate applications.
4. Enter the user’s First Name, Last Name and Email Address as needed.
To save the new user, click **Save**.

**Editing users**

**Procedure**

1. On the Users tab, select the user to be edited. The user's current information appears in the right pane.

   **Note:** You cannot change the login name for the user currently logged in. Use a different administrator login to make the necessary change.

2. Make desired changes to user information and/or group membership.

3. To save the changes, click **Save**.

**Deleting users**

**Procedure**

1. On the Users tab, select the user to be deleted. The user's current information appears in the right pane.

   **Note:** Some users are required by the system and cannot be deleted. They are marked with a lock icon. Also, you cannot delete the login name for the user currently logged in; use a different administrator login to make the necessary deletion.

2. Click the **Delete** button. A confirmation dialog opens.

3. Click **OK** to delete the user.

**Assigning users to groups**

Users can be assigned to groups on the User or Groups tab. Some users are required by the system to have specific group assignments, so their group assignments cannot be changed.

**About this task**

**Note:** Users are not required to have membership in any groups, however, without membership in the Default group (at a minimum), the user cannot log into Initiate applications.

**Procedure**

1. To grant a user membership to one or more groups:
   
   a. Select the Users tab.
   
   b. Select the desired user on the left pane.
   
   c. On the right pane, in the Group Membership box, check the groups to which the user should have membership. If desired, use the **All** button to select all groups.
   
   d. To save the changes, click **Save**.

2. To add one or more users to a group:

   a. Select the Groups tab.

   b. Select the desired group on the left pane.

   c. In the Group Members pane on the right, check the users which should have membership to this group. If desired, use the **All** button to select all users.

   d. To save the changes, click **Save**.
Removing users from groups

Users can be removed from groups on the User or Groups tab. Some users are required by the system to have specific group assignments, so their group assignments cannot be changed.

About this task

Note: Users are not required to have membership in any groups, however, without membership in the Default group (at a minimum), the user cannot log into Initiate applications.

Procedure

1. To remove a user's membership from one or more groups:
   a. Select the Users tab.
   b. Select the desired user in the left pane.
   c. In the Group Membership box on the right, uncheck the group(s) to which the user should not have membership. If desired, use the None button to deselect all groups.
   d. To save the changes, click Save.
2. To remove one or more users from a group:
   a. Select the Groups tab.
   b. Select the desired group in the left pane.
   c. In the Group Members pane on the right, uncheck the user(s) which should not have membership to this group. If desired, use the None button to deselect all users.
   d. To save the changes, click Save.

Navigating the LDAP perspective

The instructions presented here are intended as a “quick-start” to help you add users and groups specifically for the purpose of accessing Initiate applications.

About this task

The LDAP plug-in was created by Apache.org and integrated into IBM Initiate Workbench for the purpose of managing users. Full documentation on the Apache Directory Studio (contents of the LDAP perspective) is available at http://directory.apache.org/.

Procedure

1. Select Window > Open Perspective > Other from the main menu and choose LDAP from the list.
2. Click OK.

Results

When the perspective first opens, all views are empty. Next, you need to connect to the LDAP server.
Connecting to the LDAP server

To add, edit or remove users from the system, first make a connection to the LDAP server. You will need to know the host name or IP address, port and Bind DN or user and Bind password.

Procedure

1. From the LDAP menu, select New Connection.... The New LDAP Connection dialog opens.
2. Supply a name for the connection, the LDAP host and port and Encryption method, if any.
   - If you are using the internal Initiate LDAP server, the default LDAP port is 1389.
   - The internal Initiate LDAP server currently only supports No encryption.
3. Optional: Click Check Network Parameter to verify the LDAP host and port.
4. Click Next.
5. On the Authentication page, indicate the Authentication method, Bind DN or user, and Bind password.
   - For the internal LDAP server, the default Bind DN is cn(system,ou=System,ou=Users,dc=Initiatesystems,dc=com and the Bind password is system.
   - The internal Initiate LDAP server currently only supports Simple authentication.
4. Optional: Click the Check Authentication button to ensure the information entered thus far is valid.
5. Click Next.
6. Optional: For the internal LDAP server, click the Fetch Base DNs button to retrieve the base DNs from the server.
7. On the Browser options page, you can select additional connection parameters.
   - Optional: For the internal LDAP server, click the Fetch Base DNs button to retrieve the base DNs from the server.
8. Click Finish.

Results

The LDAP Browser view is refreshed and displays the DIT, Searches and Bookmarks elements. Fully expand the DIT element to access the groups and users within the base DSA-specific entry (DSE). Additional groups and users can now be added. Consult the Apache Directory Studio documentation for information on how to search groups and users within the LDAP environment and for more information on the LDAP directory structure. The Apache Directory Studio LDAP Browser help is also available in IBM Initiate Workbench by selecting Help > Help Contents.

Importing or exporting LDAP groups and users

You can use LDIF (LDAP Data Interchange Format) files to import or export user and group data from or to an external LDAP server or another Hub (if using a standalone Initiate internal LDAP server).
**Procedure**

Right-click within the LDAP Browser view to access the context menu, and select Import or Export. Refer to the Apache Directory Studio documentation for detailed instructions.

**LDAP groups**

Groups can be added directly under the ou=Groups element, or within a specific group DN (distinguished name) such as System or a new one that you create. Creating a “container” for groups is optional, but may help organize groups more easily.

All groups under the dn `ou=System,ou=Groups,dc=initiatesystems,dc=com` should not be changed. These groups are needed for normal operation of the Hub. You can create other groups that implement different behaviors if those in System do not fit your needs. These groups cannot be managed from IBM Initiate Workbench either since they are reserved.

**Optional: Create a group DN container**

You can create a “container” to hold groups within the ou=Groups element to help keep your groups organized.

**Procedure**

1. Right-click the ou=Groups element under the desired corporate DN (such as `dc=initiatesystems,dc=com`) and select New Entry.... The New Entry dialog opens.
2. On the Entry Creation Method page, select Create entry from scratch, and then click Next. If you select an existing group in step 1, you can choose to use that group as a template, which copies most of the attributes from the existing group as a starting point.
3. On the Object Classes page, select organizationalUnit in the list of Available object classes, and click the Add button to move it to the list of Selected object classes.
4. Click Next.
5. On the Distinguished Name page, provide (or check) the following values:
   - Parent – ensure that the desired parent DN is shown (such as `ou=Groups,dc=initiatesystems,dc=com`)
   - RDN – ou
   - = – type the desired container name (such as UserGroups)
6. Click Next.
7. On the Attributes page, click Finish.

**Adding new groups**

You will want multiple groups of users in order to set up permissions for application and data access.

**Procedure**

1. Right-click the desired distinguished name (DN) and select New Entry....
2. On the Entry Creation Method page, select Create entry from scratch, and then click Next.
3. On the Object Classes page, select **groupOfNames** in the list of Available object classes, and click the **Add** button to move it to the list of Selected object classes.

4. Click **Next**.

5. On the Distinguished Name page, provide (or check) the following values:
   - Parent – ensure that the desired parent DN is shown (such as ou=Groups, dc=initiatesystems, dc=com)
   - RDN – cn
   - = – type the desired group name (such as InspectorUsers)

6. Click **Next**.

7. On the Attributes page, ensure that the bolded attributes are correct, and then click **Finish**.

---

**LDAP users**

There are two main steps to adding users to the system: creating the users and assigning them to groups.

The Hub configuration utility (MADCONFIG) automatically created a system user called system. This user is for administrative/system use and should not generally be used to log into Initiate applications. However, it can be used as a template to create additional users.

**Note:** A new administrative user should be created and added to the Administrators group for all future Hub- and IBM Initiate Workbench-related administrative tasks, rather than using the provided system username. If you elect to continue using the system user name, its password should be changed.

**Adding new users**

Adding users to the system enables you to assign them to groups and give them access to your applications and data.

**Procedure**

1. Right-click on system (or another user) under the ou=Users element and select **New Entry**.

2. On the Entry Creation Method page, select **Use existing entry as template**. Check that the system (or another existing user) is shown in the pull-down list, and then click **Next**.

3. On the Object Classes page, ensure that inetOrgPerson is shown in the list of Selected object classes. If it is not, select it in the list of Available object classes, and click the **Add** button to move it to the list of Available object classes.

4. Click **Next**.

5. On the Distinguished Name page, provide (or check) the following values:
   - Parent – ensure that the desired parent DN is shown (such as ou=Users, dc=initiatesystems, dc=com)
   - RDN – cn
   - = – type the desired user name.

6. Click **Next**.

7. On the Attributes page, double-click the **uid** entry and change it to a new value.
Note: Users should be instructed to log in to Initiate applications using the cn value, not the uid value.

8. Double-click the mail entry and provide an e-mail address for this user.

9. Double-click the userPassword entry. The Password Editor dialog opens.

10. Click the New Password tab. Enter a new password. The user should be instructed to change his or her password using an employee-facing application such as IBM Initiate Inspector or Web Reports. SSHA is the recommended Hash Method. Click OK.

11. Click Finish.

Adding a password that includes multi-byte characters

The process for adding a password that includes multi-byte characters does not differ from the process of creating a new password described above except that for the new password value you will paste in a UTF-8 encoded string generated with the Encoder/Decoder tool.

About this task

Per RFC 2254, LDAP search filters containing characters above the standard ASCII character set must be encoded in UTF-8 format.

Procedure

1. From the main menu, choose LDAP > Open Encoder/Decoder.
2. Paste the multi-byte password into the ISO-8859-1 field. The Encoder/Decoder populates the other fields.
3. Highlight the UTF-8 value and copy the string into the clipboard.
4. Click OK to close the Encoder/Decoder.
5. In the LDAP Editor, left-click the user whose password you want to change.
6. In the Entry Editor, double-click the userPassword entry. The Password Editor dialog opens.
7. Click the New Password tab.
8. Paste in the UTF-8 encoded value you copied from the Encoder/Decoder. SSHA is the recommended Hash Method.
9. Click OK.

Results

The new user can now log in to the IBM Initiate Inspector or IBM Initiate Web Reports applications using the original ISO8859-1 encoded password.

Adding a user name that includes multi-byte characters

To add a user name that includes multi-byte characters, copy and paste an existing user and then set the new cn value to a UTF-8 encoded string generated with the Encoder/Decoder Tool.

About this task

Per RFC 2254, LDAP search filters containing characters above the standard ASCII character set must be encoded in UTF-8 format.
Procedure
1. From the main menu, choose **LDAP > Open Encoder/Decoder.**
2. Paste the multi-byte user name into the ISO-8859-1 field. The Encoder/Decoder populates the other fields.
3. Highlight the UTF-8 value and copy the string into the clipboard.
4. Paste the value into a separate application such as Notepad.
5. Click **OK** to close the Encoder/Decoder.
6. Right-click on a user under the ou=Users element and select **Copy Entry / DN.**
7. Right-click on the ou=Users element and select **Paste Entry.** The Select Copy Strategy page appears.
8. Choose **Rename entry and continue.**
9. For the cn value, paste in the UTF-8 encoded value you copied from the Encoder/Decoder.
10. Click **OK.** The UTF-8 encoded value appears in the LDAP Browser.
11. Left-click the new entry and edit the attribute values for uid and mail.

**Note:** Users should be instructed to log in to Initiate applications using the cn value, not the uid value.

12. Double-click the **userPassword** entry. The Password Editor dialog opens.
13. Click the New Password tab. Enter a new password. The user should be instructed to change his or her password using an employee-facing application such as IBM Initiate Inspector or IBM Initiate Web Reports. SSHA is the recommended Hash Method. Click **OK.**
14. Click **Finish.**

Results
The new user can now log in to the IBM Initiate Inspector or Web Reports applications using the original ISO/SM790000-8859-1 encoded user name. Note that within the applications the user name will appear as the UTF-8 encoded string rather than as the ISO-8859-1 encoded string.

Adding the user to a group
Once users have been created, you add them to groups in order to set their permissions.

Procedure
1. In the LDAP Browser, navigate to the group to which you wish to assign the user.
2. Right-click within the Entry Editor and select **New Attribute....** The New Attribute dialog opens.
3. On the Attribute Type page, select member as the **Attribute type.** You may need to uncheck the **Show subschema attributes only** option.
4. Click **Next.**
5. Click **Finish.** The DN Editor dialog opens.
6. Click the **Browse** button. The Select DN dialog opens.
7. Navigate to the desired user and click **OK.**
8. Click **OK** on the DN Editor dialog.
Results

*Note:* If you have added a multi-byte user name to a group, the member attribute value will include escape characters to conform to LDAP standards.

*Note:* Any users who need to manage the hub configuration via IBM Initiate Workbench should be added to the existing Administrators group (cn=Administrators,ou=System,ou=Groups,dc=initiatesystems,dc=com).

Next steps

About this task

Once groups and users are defined in the LDAP perspective (or another LDAP tool), return to the Configuration perspective to set permissions for Initiate applications and interactions. Refer to “Groups” on page 145 for details.
Chapter 10. Hub administration

Currently, the Hub administration view contains only one tab, Source Sequence Identifiers.

Source Sequence Identifiers

About this task
You can use source sequence identifiers to tell the Hub how to increment the record numbers (such as MEMRECNO) as records are added to the system.

Procedure
1. From the menu, select Window > Show View > Other.... Expand the Workbench list and select Hub Administration.
2. Click the Connect icon to establish a connection to the Hub. Supply a user name and password which has administrative privileges in the LDAP server.
3. Click the Add button. The Source Sequence Identifier Configuration dialog opens.
4. Select the Source for which you wish to configure the increment.
5. Indicate the Next Identifier to use. This number will be used as the RECNO for the next new record added to the selected source.
6. Indicate the desired Increment value. The RECNOs are usually incremented by 1.
7. Click Finish.

Results
Continue to add different Next Identifier and Increment values for other sources by repeating steps 2-6 as desired.
Chapter 11. Operational monitoring

The monitoring perspective provides a JMX-compliant application that connects to a Java Virtual Machine (JVM) to easily monitor Hub performance and resource consumption.

Note: As an alternative to using the MBean Explorer, system administrators can configure JConsole as described in the IBM Initiate Master Data Service Engine Installation Guide.

Accessing the JMX browser

About this task

You can use the JMX browser directly from IBM Initiate Workbench via the Monitoring perspective.

Procedure

1. Select Window > Open Perspective > Other... and select Monitoring. The Monitoring perspective opens with the MBean Explorer view on the left side. If the MBean Explorer view is not open, use the MBean Explorer button in the lower left corner of the screen to open it.
2. Click Connect to Hub to establish the connection to the running Hub instance. The Hub Monitoring Source Selection dialog opens.
3. Select the Hub to which you want to connect. If the desired Hub is not listed, click the Edit button and refer to "Adding a connection" on page 11.
4. Click OK. The login dialog opens.
5. Enter a user name and password which is in the Administrators LDAP group (cn=Administrators,ou=System,ou=Groups,dc=initiatesystems,dc=com), such as system/system. Refer to Chapter 9, “User management,” on page 207.
6. Click Log In. The MBean Explorer is refreshed with the list of services running on the selected Hub.

The MBean Explorer view

When the console opens, you will see the MBean Explorer view. For purposes of monitoring the Hub, the information in this section deals primarily with the MBean Explorer view and the com.initiatesystems tree.

Under the com.initiatesystems tree you will see the Master Data Engine instance selected on the connection dialog. Beneath that are four nodes—LDAP, Listeners, Logging, and ThreadPools.

If you have enabled callouts, you will also see a Callouts node. The information that appears depends on the hook points that have been established. A hook point is where callback invocation occurs during Master Data Engine processing. For more information about callouts, see Chapter 8, “Callout handlers,” on page 199.

Note: Also beneath the Master Data Engine are MBeans for Jobs and Metadata, which are for internal use only. You should not have occasion to monitor or alter those MBeans.
Within each node, is one or more MBeans. You can double-click the MBeans to view the Attributes, Operations, Notifications, and Info tabs within the right pane. (Tabs are located on the bottom of the view). For monitoring purposes, you will be primarily interested in the contents of the Attributes tab. The Info tab simply provides the MBean name and the associated Java classname.

**Listeners node**

Select Listeners to view information about Master Data Engine transactions. Within the Listeners node is the Master Data Engine port number MBean, and within that is the Interactions node (below the Port, LogIxn, and other attributes).

From the port number MBean, you can view the LastUpdateTime from the Attributes tab. From the Operations tab, you can refresh interaction statistics by selecting refreshIxnStats and clicking the `refreshIxnStats` button.

To view the activity of an interaction, expand the Interactions node. A list of all interactions appears. For descriptions of each of the available interactions, see the *IBM Initiate Master Data Service SDK Reference for Java and Web Services*.

Expand any interaction node and double-click one of the hour-delimited MBeans (or the Cumulative MBean). The Attributes tab logs the activity for each attribute for the interaction. Note that the view does not automatically change if it is in focus; click the **Refresh** button in the upper right of the pane to update the display.

Related reference:

“Interactions in the monitoring perspective”

**Interactions in the monitoring perspective**

The following table lists descriptions of the various interactions.

<table>
<thead>
<tr>
<th>Interaction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AvgBktCands</td>
<td>Average number of member candidates returned from buckets</td>
</tr>
<tr>
<td>AvgRcvSize</td>
<td>Average size of messages received by the Master Data Engine</td>
</tr>
<tr>
<td>AvgSndSize</td>
<td>Average size of messages sent by the Master Data Engine to clients</td>
</tr>
<tr>
<td>AvgTicks</td>
<td>Average number of milliseconds</td>
</tr>
<tr>
<td>MaxBktCands</td>
<td>Maximum number of member candidates returned from buckets</td>
</tr>
<tr>
<td>MaxRcvSize</td>
<td>Maximum size of messages received by the Master Data Engine</td>
</tr>
<tr>
<td>MaxSndSize</td>
<td>Maximum size of messages sent by the Master Data Engine to clients</td>
</tr>
<tr>
<td>MaxTicks</td>
<td>Maximum number of milliseconds</td>
</tr>
<tr>
<td>MinRcvSize</td>
<td>Minimum number of member candidates returned from buckets</td>
</tr>
<tr>
<td>MinSndSize</td>
<td>Minimum size of messages received by the Master Data Engine</td>
</tr>
<tr>
<td>MinTicks</td>
<td>Minimum number of milliseconds</td>
</tr>
<tr>
<td>TotBktCands</td>
<td>Total number of member candidates returned from buckets</td>
</tr>
<tr>
<td>TotBktSrchs</td>
<td>Total number of bucket searches</td>
</tr>
<tr>
<td>TotExecs</td>
<td>Total number of times this interaction was called</td>
</tr>
<tr>
<td>TotGood</td>
<td>Total number of times this interaction returned a response with no error</td>
</tr>
</tbody>
</table>
Log4j node

Select Log4j to view log settings and error log notifications. Log4j handles logging for the Master Data Engine. Within Log4j, there is a root > Appenders tree. From here you can control how logging functions in the application. The File appender governs the basic logging function for writing log messages out to a specified file. The Triggered appender defines the settings for the caching log writer, described in detail in the “Software logs and data files” chapter of the IBM Initiate Master Data Service Software Operations Guide.

Any changes to the log4j attributes you make within the Monitoring perspective will be lost the next time you restart the Master Data Engine. To make persistent changes to the logging mechanism, edit the log4j.xml file directly. Those changes will override any changes you have made via the JMX browser. The log4j.xml file is described in detail in the “Configuring Environment Variables and Settings” chapter in the IBM Initiate Master Data Service Engine Installation Guide. For additional information about log4j, refer to http://java.sun.com.

Note: Only appenders enabled within the log4j.xml configuration file will appear within the Log4j node.

Search node

A new JMX node called Search has been added under the com.initiatesystems.MPINET node. This node contains three MBean nodes: AdminApi, SearchService and SynchronizationService.

AdminApi

This MBean provides basic administrative level attributes and operations.

Attributes

- Installed - Indicates if the full text index is installed.
- MaxSearchResults - Maximum number of hits to return from a search.
- SearchTimeout - Maximum length of time a search is allowed to take in milliseconds.

Operations

- setMaxSearchResults - Sets the maximum number of hits to return from a search. Changes to this value will persist across engine restarts.
- setSearchTimeout - Sets the maximum length of time a search is allowed to take in milliseconds. Changes to this value will persist across engine restarts.
- setTempMaxSearchResults - Temporarily sets the maximum number of hits to return from a search. Changes to this value will not persist across engine restarts.
- setTempSearchTimeout - Temporarily sets the maximum length of time a search is allowed to take in milliseconds. Changes to this value will not persist across engine restarts.
SearchService
This MBean reports latency statistics for the full text search service.

Attributes
TermSearchCount - The number of executed term searches.
TextSearchCount - The number of executed text searches.
MeanTermSearchLatency - The mean term search latency in milliseconds.
MeanTextSearchLatency - The mean text search latency in milliseconds.
MaxTermSearchLatency - The maximum term search latency in milliseconds.
MaxTextSearchLatency - The maximum text search latency in milliseconds.

Operations
reset - Resets accumulators.

SynchronizationService
This MBean allows access to various operations and latency statistics for the index synchronization service.

Attributes
Running - Indicates if the synchronization service is currently running.
MaxRows - The maximum number of records to poll from the queue.
PollingInterval - The interval in milliseconds to poll for new records in the queue.
BuildBatchSize - The number of polled records to process in batch.
NodeName - The name of this node in the cluster.
SegCodeFilter - The segment code filter currently in effect based on the contents of the index configuration file.
SegAttrFilter - The segment attribute filter currently in effect based on the contents of the index configuration file.
RecStatFilter - The record status filter currently in effect based on the contents of the index configuration file.
MeanRunLatency - The mean latency observed for all steps (poll, memget, build, merge and remove) in milliseconds.
MeanPollLatency - The mean latency for polling records from the queue in milliseconds.
MeanMemGetLatency - The mean latency for executing MEMGET in milliseconds.
MeanBuildLatency - The mean latency for adding records to the index in milliseconds.
MeanMergeLatency - The mean latency for index merging in milliseconds.
MeanRemoveLatency - The mean latency for removing records from the queue in milliseconds.
MaxRecordCount - The maximum number of records polled from the queue.
MaxPollLatency - The maximum latency for polling records from the queue in milliseconds.
MaxMemGetLatency - The maximum latency for executing MEMGET in milliseconds.

MaxBuildLatency - The maximum latency for adding records to the index in milliseconds.

MaxMergeLatency - The maximum latency for index merging in milliseconds.

MaxRemoveLatency - The maximum latency for removing records from the queue in milliseconds.

MaxRunLatency - The maximum latency observed for all steps (poll, memget, build, merge and remove) in milliseconds.

**Operations**
- reset - Resets accumulators.
- start - Start the Index Synchronization Service.
- stop - Stops the Index Synchronization Service.
- setMaxRows - Sets the maximum number of records to poll from the queue.
- setPollingInterval - Sets the interval in milliseconds to poll for new records from the queue. Changes to this attribute requires a stop/start of the service.
- setBuildBatchSize - Sets the number of polled records to process in batch.

**ThreadPools node**

Through the ThreadPools node, you can monitor the service threads or context objects connected to the Master Data Engine. The ThreadPools node has two MBeans: Listener and TagManager. The Listener MBean applies to the overall working of the Hub; the Tag Manager MBean applies to custom tags based on tasks in the system.

If you have enabled callouts, you will see a Callback MBean within the ThreadPools node for monitoring callout threadpool activity. Callouts extend the Master Data Engine via custom code in the form of callback handlers that define the callout logic to communicate with third-party systems or to implement conditional security and event notification. For more information, see Chapter 8, “Callout handlers,” on page 199.

Each MBean has the same set of attributes. Clicking any of the ThreadPools MBeans opens the Attributes tab in the right pane. The ContextPoolSize is the number of Master Data Engine 'threads' that are started concurrently. Each context pool has its own connection to the database and can operate independently of the others. If you have your context pool set at 5, for example, you can send in 5 searches, gets or puts at exactly the same millisecond, and they are all processed concurrently. If 6 are sent, then the first 5 will process while the 6th waits for the next free context.

CurrentContexts is the number of context threads currently in use, and MaxContexts shows the peak number of contexts that have been in use at one time.

Each instance of the Master Data Engine is a Java object or resource to be managed—an MBean. The Info tab provides the MBean name and the Java class to
which the resource belongs. Again, the Info tab does not contain any information vital to the processing of the Master Data Engine, and mention of this tab is for information purposes only.

**Administrative actions**

If you log in with an administrative user name and password, you can bounce or stop the Master Data Engine or perform a thread dump. These actions are performed from the org.tanukisoftware.wrapper node, a sibling node to com.initiatesystems.

**Attributes tab**

Expand org.tanukisoftware.wrapper and click the WrapperManager MBean to view the Attributes tab.

**Operations tab**

To stop the Master Data Engine instance, click **Stop**. When you do so, the instance is stopped and connection to the JMX browser is terminated. You will have to restart the Master Data Engine and reconnect the JMX browser.

To restart the Master Data Engine, click **Restart**. The instance will stop and then restart. Connection to the JMX browser will be lost, but you can reconnect after waiting a few moments.

In some instances, when you are working with IBM Software Support, the support engineer may need a thread dump. By clicking requestThreadDump, the JVM thread information is output into the hub_instance_name.out log folder in the instance log directory.
Chapter 12. IBM Initiate Pair Manager

As part of the weight generation process, you ran the Generate Threshold Analysis Pairs job to create one or more .xls file(s) containing pairs of members. In order to establish threshold values for the Hub to match members accurately, you need to evaluate these sample pairs and indicate whether the members are the same or not the same. The results of the sample pair review may be used as input to the Threshold Calculator job to determine appropriate thresholds.

Users who understand the data must review each pair in the sample pairs file to determine whether the members of each pair are the same or different. There are two methods for evaluating the sample pairs: using a spreadsheet application such as Microsoft Excel, or using the IBM Initiate Pair Manager (installed separately - refer to the IBM Initiate Workbench Installation Guide). Both methods enable you to accomplish the same result: to analyze the sample pairs in order to evaluate threshold values.

The sample pair file

Within the .xls file are rows of member data grouped in pairs. Most of the columns are self-explanatory, shown in the format <Attribute Name> | <Field Label>. The specific attributes shown in the file depend on what was selected when you ran the Generate Threshold Analysis Pairs job. Other columns of importance are:

- **Same?** - indicates whether the members in each pair are the same member. One or both members of each pair should be marked with Yes, No or Maybe. This column is not displayed in the IBM Initiate Pair Manager.
- **Pair number** - numbers each pair of members sequentially.
- **Score** - the comparison score assigned by the engine. The higher the number, the greater the likelihood the two members in the pair represent the same individual.
- **Original score** - the first comparison score assigned by the engine. When members are rescored following an algorithm update, their Score often changes. The Original score, however, remains static.
- **memrecno** - the member record number in the Hub database assigned to the member.
- **Source** - the source system where the member data originated.

If you prefer to use a spreadsheet application to evaluate the sample pairs, refer to the instructions in “Generate Threshold Analysis Pairs” on page 30.

**Note:** The IBM Initiate Pair Manager supports only sample pair files created with IBM Initiate Workbench 9.7.

If the algorithm is changed after running Generate Threshold Analysis Pairs, you can rescore the sample pairs by using the **Initiate > Rescore members in sample pairs file(s)** menu. Refer to “Rescoring members in sample pair files” on page 9.
Using the IBM Initiate Pair Manager

Each pair is presented side-by-side, and each matching row of data is highlighted. This enables you to quickly determine whether the two members of each pair match, with minimal scrolling.

Procedure

1. To open the IBM Initiate Pair Manager application, click **Start > All Programs > IBM Initiate > Pair Manager 9.7.0**
2. To open the samplePairs.xls file, click the **Open** button. In the Open Sample Pair File dialog, browse to the file's location and click **Open**. The samplePairs.xls file is typically located in IBM Initiate Workbench’s `workspace\project_name` directory, but may have been saved to another path. The IBM Initiate Pair Manager screen is populated with pair data for the attributes selected in the Generate Threshold Analysis Pairs job.

Matching attributes are marked with a green checkmark in the Pair Data Labels column. The matching attribute data is highlighted in green for both members. Note that some members may have multiple attributes, such as the last name in the example below. The current last name for the member does not match, but both sources have the same last name listed as previous values for the member. Similarly, expanding the stLine1 attribute would reveal that Member #1 previously had the same address as Member #2, resulting in a match for that attribute.

**Note:** The filename you open may differ from the default (samplePairs.xls).

3. For each pair of members, evaluate each data element to determine whether the members are the same.
   - If they are the same, click the **checkmark** (is a match) button or press the Y key.
   - If they are not the same, click the **slashed circle** (is not a match) button or press the N key.
   - If it cannot be determined from the data shown, click the **question mark** (may be a match) button or press the M key.
4. To advance to the next pair, click the **Right arrow** (next) button or press the Right Arrow key. If the **Auto-advance** option is enabled, the next pair is displayed automatically. If you need to return to the previous pair, click the **Left arrow** (previous) button or press the Left Arrow key.
5. When finished evaluating pairs, click the **Save** button. You can overwrite the file or specify a new filename.

Results

If not all pairs have been evaluated, you can use the IBM Initiate Pair Manager again later to pick up where you left off, using the filtering options to easily locate the pairs that still need evaluation.

Filtering members by status

The Filter feature enables you to view pairs by status.

About this task

Select the desired option from the pull-down list to set the filter:
• To view all pairs, select Show All Pairs (the default filter). You can enable or disable the Auto-advance option when this filter is selected.
• To view only the pairs still needing evaluation, select Show Undecided Pairs Only. The Auto-advance option is always enabled when this filter is selected.
• To view only pairs marked as matching, select Show Yes Pairs Only. The Auto-advance option is always enabled when this filter is selected.
• To view only pairs marked as non-matching, select Show No Pairs Only. The Auto-advance option is always enabled when this filter is selected.
• To view only pairs which are possible matches, select Show Maybe Pairs Only. The Auto-advance option is always enabled when this filter is selected.

Selecting data columns to display
By default, the IBM Initiate Pair Manager displays all of the data fields that are in the samplePairs.xls file. If desired, you can limit the fields to show only those you are interested in comparing.

About this task
Note: The Same? column, visible when the .xls file is viewed in a spreadsheet application, is not displayable in the IBM Initiate Pair Manager.

Procedure
1. To specify the fields to show, click the Configure labels button in the upper left corner of the screen. The Label Selection dialog opens.
2. Check or uncheck the fields as desired to display or hide the corresponding data. Neither field labels nor corresponding data appear when the field name in this list is unchecked.
3. Use the Check All or Uncheck All button to check or uncheck all fields. You can then check or uncheck fields you wish to show or hide.
4. Select a field and use the Move Up or Move Down button on the right side of the dialog to move it up or down in the list. This dictates the order in which the field and the corresponding data will appear in the IBM Initiate Pair Manager. Child attributes can be reordered within their parent section, but not outside it.
5. Click OK to accept the changes and close the dialog, or click Cancel to discard the changes.

Handling large sample pair files
IBM Initiate Pair Manager is launched with a default maximum heap size of 256Mb, which is sufficient to handle a large number of pairs. Typical maximum pair counts for evaluation are approximately 5000 pairs.

About this task
If IBM Initiate Pair Manager runs out of heap space when opening a very large file, you can increase the maximum heap size by passing an argument to the IBM Initiate Pair Manager shortcut in the Start menu.

Procedure
1. Right-click the IBM Initiate Pair Manager shortcut and select Properties from the context menu.
2. In the **Target** field add `-J-XmxZZZm` after the executable name outside the quotes. `ZZZ` represents the heap size to pass. The default is set internally to 256.

For example, to set the maximum heap size to 512Mb, use this as the Target:

```
"C:\Program Files\IBM\Initiate\PairManager9.7.0\pairmanager.exe"
-<J-Xmx512m
```
Chapter 13. Initiate Inspector Configuration

With IBM Initiate Inspector version 9.7, the configuration options have been moved to IBM Initiate Workbench to make it a much simpler experience.

Before continuing, ensure that IBM Initiate Workbench version 9.7 is installed on the computer. See the IBM Initiate Workbench Installation Guide for more information.

If you had a customized configuration in previous versions, or would like to customize your configuration, everything you need is in IBM Initiate Workbench. To get started, open IBM Initiate Workbench.

Implementing configuration changes

About this task

Once configuration changes are made, the configuration needs to be saved and uploaded to the Hub in order to be visible in the IBM Initiate Inspector.

Procedure

1. To save the configuration, click the Save icon on the toolbar, or press Ctrl+S on the keyboard.
2. To upload the configuration, refer to “Deploying a Hub configuration” on page 17. Keep in mind that other Hub configuration options are deployed with the same process.
3. Restart IBM Initiate Inspector to view the changes.

IBM Initiate Inspector Configuration Editor

Everything you will need to configure IBM Initiate Inspector can be found in the Inspector Configuration Editor within IBM Initiate Workbench. Switch to the Inspector Configuration Perspective, and then open the inspector.arm file shown in the Navigator view. The Inspector Configuration Editor opens. For more information on switching perspectives, see Chapter 1, “IBM Initiate Workbench basics,” on page 1.

On the left of the page, the options for configuration are listed:

- **Attribute Display** - This page determines the Attribute Labels, Patterns and display order per Member Type.
- **Custom Task Summary** - Use this page to configure attributes to display per member type.
- **General Preferences** - Use this page to configure basic page elements such as page size, date formatting, maximum search results, etc.
- **Member and Entity** - Use this page to view previously configured Member/Entity types, Composite Views, and default Hierarchy Relationships (if any).
- **Search Forms** - Use this page to determine what fields you want to display in the IBM Initiate Inspector search forms.
Search Results - Use this page to determine which fields to display in the IBM Initiate Inspector results screens.

The Inspector Configuration Editor works together with the Hub Configuration. For more information, see Chapter 4, "Configuration editor," on page 89.

Attribute Display page

Use this page to customize the attribute display for various member types and/or add new attributes and attribute types to display in IBM Initiate Inspector.

Customizing attributes

About this task

Use the Member Type drop-down list to select a member (record) type.

The Attributes grid appears below the drop-down list. The attributes are in the order in which they will display in Inspector.

Procedure

1. To change the attribute order, select an attribute, and then click the up or down icons in the center of the screen.
2. If an attribute has been deleted from a member type via Hub Configuration in IBM Initiate Workbench, the red X icon is enabled. To delete the attribute from the Inspector view, click the red X icon.

Note: Within the Attributes grid, when you click on an Attribute an Attribute Type on the right is highlighted.

Customizing the attribute pattern

About this task

By default, the Use Default Type Pattern? checkbox is selected. If you de-select that option, the Attribute Pattern box becomes active.

To insert a field:

Procedure

1. Click Insert Field. The Available Fields box opens.
2. Select a Field, and then click OK. The field is added to the Attribute Pattern box.

To insert Optional tags:

Procedure

1. Place the cursor in the Attribute Pattern box where you want the Optional tag to appear.
2. Click Insert Optional Tag.

Customizing attribute types

About this task

The Attribute Type grid displays the Attribute Types currently configured for IBM Initiate Inspector.
Procedure
1. To add an attribute type to display, click the Add icon. The Select a segment box opens.
2. Click OK. The Attribute Type is added to the grid.

Customizing attribute type default patterns

To insert a field:
Procedure
1. Click Insert Field. The Available Fields box opens.
2. Select a Field, and then click OK. The field is added to the Attribute Pattern box.

To insert optional tags:
Procedure
1. Place the cursor in the Attribute Pattern box where you want the Optional tag.
2. Click Insert Optional Tag.

Results

For MEMDATE and MEMATTR segments, leave the patterns blank so that the format will default to the specified patterns on the General Preferences page.

To implement your changes:
Procedure
1. Save your changes.
2. From the menu, select Initiate > Deploy Hub Configuration to commit your changes.
3. Restart IBM Initiate Inspector to see the changes in the application.

Results

Each time you Deploy, all changes made in IBM Initiate Workbench are deployed, not just the Inspector Configuration changes. The engine is suspended and resumed with each deployment. For more information, see Chapter 2, “Managing projects and Hub connections,” on page 11.

Custom Task Summary Page

About this task

Use this page to configure the attributes to display per member type.

Select Member Type
Procedure
1. To select a member type, use the drop-down list to select a pre-configured member type. If you would like to add a member type, see “Member types” on page 89.
2. To add a task resolution formatting for another member type, click the Add icon.
3. From the Select a member type dialog, click a member type, and then click OK. To remove the task resolution formatting for the selected member type:
4. Click the Remove icon. The member type is removed from the drop-down list.
Choosing attributes to display for custom task summaries

About this task

Use this page to select which fields you want to display on the Custom Task Summary within Inspector.

- **Currently Hidden:** This column lists all of the attributes/fields that are currently hidden in the Custom Task Summary in Inspector. Use the arrows to move selected attributes/fields to the Currently Displayed column.

- **Currently Displayed:** This column lists all of the attributes/fields that will display in the Custom Task Summary in Inspector. Use the arrows to change the order or move selected attributes/fields to the Currently hidden column.

Use the Field Pattern section to determine the display values on the Custom Task Summary within Inspector.

Procedure

1. Click **Insert Field**. The Available Fields box opens.
2. Select a field, and then click **OK**.
3. Click **Insert Optional Tag** to designate the field as Optional within Inspector.

To implement your changes:

4. Click **Save**.
5. Refer to “[Deploying a Hub configuration](#)” on page 17 for detailed instructions on committing the changes to the Hub.

6. Restart IBM Initiate Inspector to see the changes in the application.

General Preferences page

As stated previously, IBM Initiate Inspector will run if you choose to accept all of the defaults found on the General Preferences page. You can change any or all of the defaults to suit your needs.

**Maximum Search Results**

Change this setting to increase or decrease the number of search results that will display in IBM Initiate Inspector. The default setting is 1000.

**Page Size**

This setting controls the page size within Inspector. The default is 20. If there are more than 20 results, tasks, and so forth on screen, you will be able to go to the next page to view them.

**Rule Violations Page Size**

This setting controls the page size in the Potential Issues grid when viewing Relationship tasks. The default is 5; if more than 5 Potential Issues are present, the results will continue to another page.

**Relationship View Cloud Threshold**

When the number of relationships exceeds the threshold set during configuration, a cloud displays in the Relationship View. The default is set to 5.

**Hierarchy View Cloud Threshold**

When viewing hierarchies, this setting will determine how many children will show on screen. The default is 100. If there are more than 100, a cloud appears in the hierarchy tree on the left of the Hierarchies tab. If you click on the entity with the cloud, the children of that entity display in the Details tab, with the ability to page through the results.
**Ellipse Length**
This setting determines the number of characters to display before displaying an ellipsis on the tabs within Inspector. For example, if you search for Patty Countryman and then click Inspect, you will see her name at the top of the tab with ellipsis. The default is 15. This means if the name goes more than 15 characters, the ellipsis appears.

**Tasks to Show in Task Resolution Screen**
The default for this setting is 25. Change the number to increase or decrease the number of tasks to display on the Task Resolution screen.

**Tasks to Show in Member Grid**
The default for this setting is 5. Change this number to increase or decrease the tasks displayed in the Member (Record) grid within Inspector on the Task Resolution screen.

**Default Task Resolve Status**
Use this field to determine your default task resolve status. The default is set to Resolved.

**Landing Page**
Use the drop-down to select the page you want IBM Initiate Inspector to open with. Valid options are Search, Inbox, or Resolve.

**Enable "Add a Record" module checkbox**
Select this check box to display the Add A Record button in IBM Initiate Inspector.

**Show user**
Clear the check box to hide the user name. By default, IBM Initiate Inspector displays the user name.

**Show host/port**
Clear the check box to hide the user’s host and port information. By default, IBM Initiate Inspector displays the host and port information.

**Automatically set Potential Overlay Task Status on save**
Select the check box to indicate that when the user saves a Potential Overlay Task, the task status should be set to the same status as that specified for Default Task Resolve Status. If the check box is not selected, attempting to save a Potential Overlay Task without explicitly setting the task status causes an error.

**Date Formatting**
This section offers four options for formatting the date and timestamp display in IBM Initiate Inspector.

- **Use localized style for date display** - Once you select this option, use the drop-down list to select the date style for the display. The options are Short, Medium, Long, or Full. An example of each style appears below the drop-down when selected.

- **Use pattern for date display** - Select this option to change the pattern for the date display. For example, type MM.dd.yyyy to display the date with periods instead of slashes.

- **Use localized style for timestamp display** - Once you select this option, use the drop-down list to select the date style for the display. The options are Short, Medium, Long, or Full. An example of each style appears below the drop-down when selected.
Use pattern for timestamp display - Select this option to change the pattern for the timestamp display. For example, delete the ss from the pattern and seconds.

Member and Entity page

Use this page to determine the member and entity type settings and add fields and label patterns.

Member Type and Entity Type Settings

The grid displays the Member/Entity types that have been configured via IBM Initiate Workbench. For more information on configuring Member and Entity types, see "Member types" on page 89. Some of the options on the screen are only available once you have added or deleted a member/entity type within the Hub Configuration.

From this grid, you can see the Member/Entity types that have already been configured, the Composite View for each, and the Default Hierarchy Relationship (if any).

Just as Member and Entity types are configured within IBM Initiate Workbench, so are the Composite Views and Hierarchy Relationships. For more information, see "Composite Views" on page 97, and "Relationship types" on page 122.

In the Inspector Configuration Editor, if there are additional Composite Views configured, you will see a drop-down in the Composite View field. This allows you to assign a different composite view to the entity or member type.

In the Default Hierarchy Relationship column, the drop-down appears if a hierarchical relationship exists based on the relationship types defined per entity type in IBM Initiate Workbench.

Note: It is recommended not to edit the default MMCA and EMCA composite views. It is better practice to create new composite views. For information on creating composite views, see "Composite Views" on page 97.

On the right side of the grid, you will see a green plus sign icon, and a red X icon. The red X icon is only enabled if the member/entity has been deleted via hub configuration. When active, the plus sign is used to add configurations for any missing member/entity types. The red X icon allows you to remove configuration for the selected entity/member type.

All valid entity types/member types must have a label and a view defined.

Setting the label pattern

About this task

Use this section to determine the properties for the display labels within Inspector.

Procedure

1. In the Attribute field, use the drop-down to select an attribute.
2. Click Insert Field. The Available Fields box opens.
3. Select a field, and click OK.
4. Click Insert Optional Tag to insert an Optional Field.
5. Click Save to save changes.
Results

Be sure to complete all of the necessary Attributes in the Label Patterns section. Check the Problems view and resolve all errors so that the labels will display correctly within IBM Initiate Inspector. For more information, see “Problems view” on page 6.

Search Forms page

About this task

Use this page to determine which fields to be included in the Search pages. The Search forms can vary between entity types.

Selecting the entity type

About this task

Use the drop-down list to select a pre-configured entity type. If you would like to add an entity type, see “Entity types” on page 93.

Adding a search form for another entity type

Procedure

1. To add a search form for another entity type, click the Add icon.
2. The Select an entity type box opens.
3. Select an entity type and the click OK. The Currently Displayed grid opens with the list of configured Attributes/Fields for the selected entity type.
4. From the Currently Displayed list, highlight the Attributes/Fields that you want to hide/show, and then use the arrow icons in the center of the screen to move them to the proper column. You can also re-order the attributes using the up and down arrow icons.
5. Click Save.

Removing a search form for an entity type

Procedure

1. Use the drop-down list to select the attribute to remove the search form.
2. Click the Remove icon. The Search form is removed for the selected entity type.

Choosing Fields to Display on Search Page

About this task

Use this page to select which fields you want to display on the Search forms within Inspector.

- **Locale**: Use the drop-down list to select the Locale. English is the default. When you switch locale to a different language/country, type the labels in that language. If you do not supply a label in the selected language, English is used.
- **Currently Hidden**: This column lists all of the attributes/fields that are currently hidden in the Search form in Inspector. Use the arrows to move selected attributes/fields to the Currently Displayed column.
- **Currently Displayed**: This column lists all of the attributes/fields that will display in the Search form in Inspector. Use the arrows to change the order or move selected attributes/fields to the Currently hidden column.

Click Save to save all changes. Refer to “Deploying a Hub configuration” on page 17 for instructions on implementing the changes to the Hub.
Search Results page
About this task

Use this page to select the columns to include in the Search Results.

Select results entity type
About this task

Use the drop-down list to select a pre-configured entity type. If you would like to add an entity type, see “Adding an entity type” on page 95.

Adding a search result form for another entity type
Procedure
1. To add a search results form for another entity type, click the Add icon.
2. The Select an entity type box opens.
3. Select an entity type and click OK. The Currently Displayed grid opens with the list of configured Attributes and Column labels for the selected entity type.
4. From the Currently Displayed list, highlight the Attributes and Column Labels that you want to hide/show, and then use the arrow icons in the center of the screen to move them to the proper column. You can also re-order the attributes using the up and down arrow icons.
5. Click Save.

Choosing columns to display in search results
About this task

Use this page to select which fields you want to display on the Search Results forms within Inspector.

- **Locale**: Use the drop-down list to select the Locale. English is the default.
- **Currently Hidden**: This column lists all of the attributes and column labels that are currently hidden in the Search Results form in Inspector. Use the arrows to move selected attributes and column labels to the Currently Displayed column.
- **Currently Displayed**: This column lists all of the attributes and column labels that will display in the Search Results form in Inspector. Use the arrows to change the order or move selected attributes and column labels to the Currently hidden column.

Procedure
1. Once you have made changes, click Preview. The Select sample data to view box opens.
2. Complete the fields and click OK. The Preview opens.

**Note**: Make sure the memrecnos you select are associated with the member/entity type being configured in the editor.

Field Pattern
About this task

Use this section to determine the display values on the Search Results within Inspector.
**Procedure**
1. Click **Insert Field**. The Available Fields box opens.
2. Select a field, and then click **OK**.
3. Click **Insert Optional Tag** to designate the field as Optional within Inspector.

**Setting a minimum score**
**Procedure**
1. From the Search Results page, open the Properties view. (To open this view go to: **Window > Show View > Other...** and then select **Properties** from the **General** folder.)
2. From the Properties view, type a minimum score in the Value field.
   The default minimum score is 0. This does not mean the minimum score is set to 0.0. It means there is no minimum score; all scores are considered, including negative scores (for Potential Overlay tasks). Scores should be entered using decimals, for example, 8.0 and not 80.
3. Click **Save**.
Chapter 14. IBM Initiate Flexible Search

Introduction

This set of topics provides information and instruction on configuring IBM Initiate Flexible Search using IBM Initiate Workbench.

For details on the architecture of IBM Initiate Flexible Search or how to use it to search with the API or IBM Initiate Composer, consult the IBM Initiate Flexible Search User's Guide.

Creating an index configuration

In IBM Initiate Workbench, you create a IBM Initiate Flexible Search index configuration from the Initiate menu. An index configuration enables you to specify options for the IBM Initiate Flexible Search index, such as date formats and synonym sets.

Procedure

1. From the IBM Initiate Workbench menu, select Initiate > Create Flexible Search Configuration.
2. In the Select a project dialog, select the Initiate project for which you want to create the new index configuration and click OK. If the Hub configuration contains errors, you can cancel the operation to fix the errors first. Proceeding without correcting configuration errors might cause unexpected behavior.
3. Select whether you want to create the index configuration based on one of the following options:
   - all attributes in the member model (all member types are included)
   - only attributes from a specified algorithm (only the member type associated with the algorithm is included). Keep in mind that:
     - if an attribute is used in a comparison function that uses a phonetic search query, the corresponding index field will have its Use IDENTAPHONE phonetic property automatically set to true.
     - if the comparison function associated with an attribute has an EQUI String code setting (nickname), a synonym set will also be created for that nickname file.
4. Click Finish.

Results

The new index configuration is created, populated with the project's hub configuration metadata and automatically opened in the IBM Initiate Workbench Editor pane. To open it later, open the flexsearch folder in the Navigator pane and double-click the flexsearch.fsm file. The editor opens with the index configuration data loaded.

What to do next

Set the IBM Initiate Flexible Search configuration options by Configuring the search index on page 242.
Configuring the search index

You configure the index for IBM Initiate Flexible Search by editing the index configuration file.

Procedure

1. In IBM Initiate Workbench, double-click the flexsearch.fsm file in the flexsearch folder in the Navigator view to edit the index configuration.
2. In the Flexible Search Configuration editor, click the Index Fields icon to open the Index Fields view.
3. In the Index Fields view, review the list of existing attributes and fields for which index fields will be automatically configured.
4. To add a new index field entry, click Add.
   a. Provide a name for the new index field entry.
   b. Set the values in the Properties view.
   c. Add attributes and fields as needed by editing the index field entry.
5. You can change or delete attributes and fields for an index field entry by clicking Edit or Remove.
6. Edit the properties for an index field entry by setting the values in the Properties view.
7. When you finish specifying index fields, click the General Configuration icon to access the general options.
8. Select the desired values for each field or option.
9. When you finish specifying general options, click the Search Tuning icon to access the tuning options.
10. Select values for each field or option.
11. Save the configuration.

What to do next

Deploy the Flexible Search Configuration to the Hub by using the Initiate > Deploy Hub Configuration menu or the Deploy Hub Configuration job, then use the Index by Members job in IBM Initiate Workbench to create the actual search index. Refer to the Index by Members job in the IBM Initiate Workbench User’s Guide for more information.
You select the General Configuration options on the General Configuration tab. Clicking the General Configuration icon on the IBM Initiate Flexible Search Configuration editor opens the General Configuration view.

**Index fields**

Used to index member attributes and fields. The index configuration file requires one or more Index Fields.

**Search tuning** on page 245

You select the search tuning options on the Search Tuning tab. Search tuning enables you to specify fine-tuning options for the search index, such as score boost.

**Index by Members job** on page 82

This job enables you to run the Index by Members utility from IBM Initiate Workbench, which indexes member data stored in the IBM Initiate Master Data Service.

---

### Index configuration file

The index configuration file, flexsearch.fsm, contains the search index metadata needed to execute searches with IBM Initiate Flexible Search.

The Create Flexible Search Configuration utility, accessed from the IBM Initiate Workbench Initiate menu, creates the index configuration file and opens it in the Flexible Search Configuration editor automatically.

You should use only the Flexible Search Configuration editor in IBM Initiate Workbench to edit the index configuration file; do not attempt to edit the file with any other editor.

### Index fields

Used to index member attributes and fields. The index configuration file requires one or more Index Fields.

The Index Fields button in the IBM Initiate Workbench editor opens the Index Fields view which displays the values currently specified for the index configuration. The Index Fields consist of the following properties:

#### Index Field Name

Each index field has a name associated with it. The name is used at query time to refer to the field. This property is required.

**Note:** If the index field name includes spaces, it must be quoted when used in the API.

#### Attribute

The Attribute specifies which member attributes will be indexed. One or more attributes are required. The Attribute column cannot be edited directly. It is populated depending on the fields selected.
Member

The Member property shows the member type for the selected fields. This property is not editable directly. The information is displayed in the grid for informational purposes.

Fields

The fields specify the actual attribute fields that will comprise the index. For each attribute selected, you may choose one or more fields to use in the index. If an attribute is identified in the fsm file, it must have at least one field property.

Indexed

By default all Index Fields are indexed, which means they are searchable. If you do not want an Index Field to be searchable (for example, because it is a stored field) set this property to false.

Name

The Name property is identical to the Index Field Name in the grid. You can set or change it in the grid or in the Properties view. This property is required.

Stored

By default, index fields are not stored. If you want the value of a field to be available from the full-text index, set Stored to true. See the Java API documentation for the details of accessing stored fields.

Use NORMPHONE phonetic

By default, index fields are not phonetically encoded. If you want to be able to search for tokens in a field phonetically, set Use IDENTAPHONE phonetic to true. See the Inexact Queries topic to learn how to search phonetically.

Type

Index Fields have an associated type which controls the way the text is interpreted and indexed. The available field types are:

STRING
The STRING field type treats the text as a sequence of one or more tokens. Tokens consist of a sequence of letters and/or numbers.

NUMERIC
The NUMERIC field type treats the text as a sequence of numbers. Numbers contain only the digits 0-9. All other characters are removed and treated as token separators. The NUMERIC field type is useful for attributes like phone numbers where the data may consist of more than one numeric value.

INTEGER
The INTEGER field type treats the text as a single decimal number. The number can have an optional leading sign (+ or -). The allowable range is 64 bits (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807). Any text that does not fit this pattern is ignored (not indexed).

FLOAT
The FLOAT field type treats the text as a single floating-point number. The
format of the number consists of an integral part, fractional part and exponent. Exponents use the letter 'e'. The allowable range is 4.9e-324 to 1.7976931348623157e308. Any text that is not formatted as a floating-point number is ignored (not indexed).

**DATE**

The DATE field type treats the text as a single date value. The default format for the DATE type is "month/date/year." Other formats can be configured using the dateFormat element. Any formats configured with the dateFormat element will replace the default format.

**Related reference:**

You select the General Configuration options on the General Configuration tab. Clicking the General Configuration icon on the IBM Initiate Flexible Search Configuration editor opens the General Configuration view.

**General configuration**

You select the General Configuration options on the General Configuration tab. Clicking the General Configuration icon on the IBM Initiate Flexible Search Configuration editor opens the General Configuration view.

**Parameters**

The General Configuration icon in the IBM Initiate Workbench Flexible Search Configuration editor opens the General Configuration view which displays the values currently specified for general configuration parameters.

**Enable Default Index Field**

By default, a master field named MasterField is created in the index. The MasterField contains the text of all other fields. It can be used to search for tokens that might appear in more than one index field.

**Record Status Filter**

By default, only active records are indexed. See the IBM Initiate Master Data Service documentation for a definition of all valid record status values.

**Date Formats**

Use this field to specify valid formats for the DATE field type. The index configuration file can have zero or more Date Formats. The syntax supported is the same as that described in the java.text.SimpleDateFormat class.

**Synonym Sets**

Use this field to specify synonyms to use with a synonym query. The index configuration file can have zero or more Synonym Sets values. A Synonym Set consists of the following:

**Set Name**

This value specifies the name of the synonym set. This name is used in a query to refer to the synonym set. When specifying a synonym set, a value for the Set Name is required.

**String Name**

This value specifies the desired string value file configured in the Hub for a string code whose type is EQUI.

**Search tuning**

You select the search tuning options on the Search Tuning tab. Search tuning enables you to specify fine-tuning options for the search index, such as score boost.
Parameters

The Search Tuning icon in the IBM Initiate Workbench Flexible Search
Configuration editor opens the Search Tuning view, which displays the values
currently specified for tuning search parameters.

Score Boost
During a synonym search, the synonym Score Boost increases the score of
tokens that match the search term exactly. When Score Boost is used, synonyms
will score lower than exact matches. Synonym Score Boost is a floating-point
number. By default the Score Boost is 0.00. Valid values are 0.0 to 100.0.

Like Term Matching
Specifies the minimum similarity for 'like' search queries. This optional element
is a floating-point number between 0 and 1 where
• 1=terms must match exactly
• 0=terms need not match at all

By default, Like Term Matching is 0.5. Valid values are 0.0 to 1.0.

Initial Prefix Match
Specifies the number of initial characters that must match exactly for a 'like'
search query. By default, the Initial Prefix Match is 1. Valid values are 1 to 10.
Appendix A. Algorithm function descriptions

The Master Data Engine uses algorithms to analyze member records whenever a search, update, bulk cross match, or weight generation is executed. The algorithms use functions to direct the processes of member data standardization, bucketing of candidates (members), and comparison of members.

The Composite Algorithm Functions feature enables IBM Initiate Workbench to display plain English function names for standardization, bucketing and comparison functions. Information about enabling this feature is found in the Algorithm Editor topics (“Classic vs. composite algorithm function names” on page 153). One or more properties are required to configure the composite algorithm. The tables in this topic provide mapping information between the composite and classic functions and the IBM Initiate Workbench properties used to configure them.

Attention: The functions listed in this chapter might not be a complete list.

Table 67. Composite-to-classic standardization function Map

<table>
<thead>
<tr>
<th>Composite Function</th>
<th>Classic Name</th>
<th>Type/Region</th>
<th>Include source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract Code</td>
<td>ABSCODE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>DATE1</td>
<td>Standard</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AGE</td>
<td>Age</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GRDATE</td>
<td>Fixed Range</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATE2</td>
<td>Partial</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>ATTR</td>
<td>Alphanumeric</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATTRA</td>
<td>Alphabetic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATTRN</td>
<td>Numeric</td>
<td></td>
</tr>
<tr>
<td>Phone</td>
<td>PHONE1</td>
<td>North America</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PHONE2</td>
<td>Full</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INTPHONE</td>
<td>International</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AUSPHONE</td>
<td>Australia</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PHONEEND</td>
<td>Last Digits</td>
<td></td>
</tr>
<tr>
<td>Postal Code</td>
<td>USZIP</td>
<td>United States</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNZIP</td>
<td>Canada</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAZIP</td>
<td>North America</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UKZIP</td>
<td>United Kingdom</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INTZIP</td>
<td>International</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AUSTPOST</td>
<td>Australia</td>
<td></td>
</tr>
</tbody>
</table>
Table 67. Composite-to-classic standardization function Map  (continued)

<table>
<thead>
<tr>
<th>Composite Function</th>
<th>Classic Name</th>
<th>Type/Region</th>
<th>Include source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>CXNM</td>
<td>Company</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CJKCXNM</td>
<td>Company Unicode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PXNM</td>
<td>Person</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UCFREQXNM</td>
<td>Person Unicode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BXNM</td>
<td>Provider</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>USADDR</td>
<td>United States</td>
<td></td>
</tr>
<tr>
<td></td>
<td>USADDR2</td>
<td>United States - Expanded</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UKADDR2</td>
<td>United Kingdom</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAADDR2</td>
<td>North America</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INTADDR2</td>
<td>International</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNADDR</td>
<td>Canada</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNADDR2</td>
<td>Canada - Expanded</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UCSFREQADDR</td>
<td>Universal Character Set</td>
<td></td>
</tr>
<tr>
<td>Email Address</td>
<td>EMAIL</td>
<td>Race</td>
<td></td>
</tr>
<tr>
<td>Biometric</td>
<td>RACE</td>
<td>Race</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EYECOLOR</td>
<td>Eye Color</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HAIRCOLOR</td>
<td>Hair Color</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HEIGHT</td>
<td>Height</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WEIGHT</td>
<td>Weight</td>
<td></td>
</tr>
<tr>
<td>Geocode</td>
<td>GEO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identity</td>
<td>IDENT1</td>
<td>Alphanumeric</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>IDENT1A</td>
<td>Alphabetic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IDENT1N</td>
<td>Numeric</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IDENT2</td>
<td>Alphanumeric</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td>IDENT2A</td>
<td>Alphabetic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IDENT2N</td>
<td>Numeric</td>
<td></td>
</tr>
<tr>
<td>Multiple Attribute</td>
<td>MULTIDIM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Passthrough</td>
<td>PASSTHRU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 68. Composite-to-classic comparison function mapping

<table>
<thead>
<tr>
<th>Composite Name</th>
<th>Classic Name</th>
<th>Type</th>
<th>Kind</th>
</tr>
</thead>
<tbody>
<tr>
<td>String Pair</td>
<td>ATTR2S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address &amp; Phone</td>
<td>AXP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 68. Composite-to-classic comparison function mapping (continued)

<table>
<thead>
<tr>
<th>Composite Name</th>
<th>Classic Name</th>
<th>Type</th>
<th>Kind</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>PXNM</td>
<td>Person</td>
<td></td>
</tr>
<tr>
<td></td>
<td>QXNM</td>
<td>Person</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CXNM</td>
<td>Company</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CXNM_CS</td>
<td>Company</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BXNM</td>
<td>Provider</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>DATE</td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATE2</td>
<td>Date or Age</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DOBA</td>
<td>Date or Year</td>
<td></td>
</tr>
<tr>
<td>Edit Distance</td>
<td>DR1D1A_SRC</td>
<td>Simple</td>
<td>1 Role, 1 Dimension with source check</td>
</tr>
<tr>
<td></td>
<td>DR1D1B_SRC</td>
<td>Quick</td>
<td>1 Role, 1-4 Dimensions</td>
</tr>
<tr>
<td></td>
<td>DR1D1C_SRC</td>
<td>Full</td>
<td>2 Role, 1 Dimension</td>
</tr>
<tr>
<td></td>
<td>DRxDyA *</td>
<td>*A corresponds to Simple,</td>
<td>3 Role, 1 Dimension</td>
</tr>
<tr>
<td></td>
<td>DRxDyB *</td>
<td>B corresponds to Quick and</td>
<td>4 Role, 1 Dimension</td>
</tr>
<tr>
<td></td>
<td>DRxDyC *</td>
<td>C corresponds to Full</td>
<td></td>
</tr>
<tr>
<td>Equivalency</td>
<td>EQVD</td>
<td>Alphanumeric</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EQVN</td>
<td>Numeric</td>
<td></td>
</tr>
<tr>
<td>False Positive Filter</td>
<td>FPF</td>
<td>Expanded=false</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FPF2</td>
<td>Expanded=true</td>
<td></td>
</tr>
<tr>
<td>Geocode</td>
<td>GEO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Height &amp; Weight</td>
<td>HXW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>US ZIP code</td>
<td>USZIP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 69. Composite-to-classic bucketing function mapping

<table>
<thead>
<tr>
<th>Composite Name</th>
<th>Classic Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>ADDR2</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>ATTR</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>PXNM</td>
<td>Person</td>
</tr>
<tr>
<td></td>
<td>CXNM</td>
<td>Company</td>
</tr>
<tr>
<td></td>
<td>BXNM</td>
<td>Provider</td>
</tr>
<tr>
<td>Date</td>
<td>DATE</td>
<td></td>
</tr>
</tbody>
</table>
Standardization functions

The standardization process “cleans” or transforms member data into a format that makes attribute comparison by the Master Data Engine easier.

Normally, standardization of data means capitalization of all alphabetic characters, removal of punctuation, anonymous value checks, and data ordering and validation. After being standardized, the data is stored as the comparison components of the derived data and used in the generation of the bucketing data. The standardized data is stored in the mpi_memcmpd table only. It is never overwritten in the member files, so the original member data is always preserved. For example, a phone number might be entered into a source as 232-123-4567. The standardization routine might strip the dashes and the area code and format the number as 1234567. That number is stored in mpi_memcmpd while the original number stored in the mpi_memphone table remains 232-123-4567.

The derived data comparison string created by the standardization process is stored in the cmpval field in mpi_memcmpd and has the following properties:

- Attributes are delimited by carats (^)
- Tokens within the attributes (for example, last name, first name) are delimited by colons (:)
- Strings are less than 256 characters in length. If a string is greater than 256 characters, a second line in the mpi_memcmpd table is created.

An example of how the data looks in the cmpval field is:
```
PUBLIC:JOHN:Q::^SSA:482891822^19681024^^M^85545:1043_WEASYST+5556060
```

The maximum number of standardization roles (std roles) an algorithm can have is 32. A standardization role serves as a placeholder for standardization roles that are mapped to specific attributes. The stdroles further assist in mapping the standardization functions to comparison functions (see “Comparison functions” on page 289).

Many standardization functions can take a minimum and maximum length argument. This argument means that the input string must contain the number of characters specified as the min and max. A string with less than the minimum length is treated as an anonymous value. When a string contains more characters than the maximum length, the first number of maximum characters are used. For example if a function has a maximum length of 7, the input string of “123456789” would be truncated to “1234567”. Unless otherwise noted, this process is true for all standardization functions that use a min and max length.

Standardization functions can be grouped by the following categories:

- Address
- Attribute
- Biometric
- Date
- Geographic
- Identifier
- Name
- Phone
- Miscellaneous
The mpi_stdfunc table stores the descriptions and names of your standardization functions.

Before reviewing the various functions, there are two filtering concepts you will want to be familiar with: anonymous values and equivalent values.

**Anonymous values**

Anonymous (ANON) values are a way by which the Hub filters out invalid input values.

These anonymous values are not used in the comparison or bucketing functions. There are two ways in which an input value can be an ANON value:

1. **Table lookup** - The mpi_stranon table stores a list of ANON values. If the ANON table is specified in mpi_dvdxstd, this input value is searched for in the mpi_stranon table. If the value is found, it is treated as ANON and is not used in comparison or bucketing routines. ANON string codes and their values are added to a Hub configuration by using the Strings tab in the Hub configuration editor. These string codes are associated with specific standardization functions with the algorithm editor.

2. **Boundary conditions** - Some functions use minimum (min) and maximum (max) values to define the allowable boundaries for the input values. If these conditions are not satisfied, the input is treated as ANON. Likewise, missing input values are also treated as ANON values.

**Anonymous value examples:**

ANON values for NAME: BABYBOY, UNKNOWN
ANON values for SSN: 999999999
ANON values for phone number: 000 0000

The ADDR2 functions (for example, CNADDR2, USADDR2, and so on) allow multi-token anonymous addresses. An ADDR2-MANON strcode must be defined in the mpi_stranon table. The value to add to mpi_stranon can be obtained from the mpi_memcmpd table for a given member. You do NOT specify ADDR2-MANON as an anonstrcode. The anonstrcode is used for single-token anonymous values only.

ADDR2 standardization functions allow the # (hash) sign as a unit marker in addresses if the dvdarg ALLOWHASH is used. The # is stored as part of the final cmpval string. Otherwise, the # is considered punctuation and stripped from the output.

**Equivalent values**

Equivalent (EQUI) values are another way by which the standardization function input values are filtered.

The mpi_strequi table contains the allowable equivalent values. When specified, if the input value is found in the table, it is standardized and used in the comparison and bucketing routines. If the input value is not found in mpi_strequi, it is treated as an ANON value.

**Equi value examples:**

EQUI values for Gender: MALE, FEMALE, M, F
Address standardization functions (street, postal, ZIP code, and email)

Address formats, whether it be mailing, location, or email, often vary from region to region.

Address standardization functions are available for mailing or location addresses, ZIP codes, international addresses and postal codes, and email addresses.

**AUSTPOST**
AUSTPOST standardizes an Australian postal code to a 4-digit string.

*Output Type*
numeric

*Fldargs*
zipcode

*MinFldArgs, MaxFldArgs*
1,1

*Number of standard roles*
1

*MinLength, MaxLength*
4,4

*CMAP strcode*
CMAP

The standardization process is:
1. If there is a CMAP table specified, the CMAP processing is done first.
2. The non-digits are filtered out of the attribute. If the attribute is empty, it is treated as ANON.
3. If the length is less than MinLength, the attribute is treated as ANON.
4. If the length is greater than the max length, the attribute is truncated to the max value. If the value is longer than the max length, the first max characters are used.
5. If there is an ANON table specified, the attribute is checked to see if it exists in the table and if so, it is treated as ANON.
6. If there is an EQUI table specified, and if the attribute is specified in the table, it is treated as ANON.

**CNADDR**
This function is used to standardize a Canadian address.

The CNADDR function works in the same manner as USADDR. ("USADDR" on page 259)

**CNADDR2**
The CNADDR2 function standardizes addresses by dividing them into the subcomponents of Canadian region and postal codes.

CNADDR2 follows the same step as USADDR2 ("USADDR2" on page 261), except for the handling of the postal code. In USADDR2, the ZIP code and two additional arguments, for state and city, are used. The state and city are used in comparison when the ZIP code is not available. For CNADDR2, the function takes a postal code and two additional arguments for Canadian region and city. If the postal code
is not available, then the region and city are used in comparison. A dvdarg of ALLOWHASH can be used to prevent any # (number sign) in the address from being treated as punctuation.

**Output Type**
- any

**Fldargs**
- stline1, stline2, stline3, stline4 [city, state, postalcode]

**MinFldArgs, MaxFldArgs**
- 1, 7

**dvdargs**
- ALLOWHASH

**Number of standard roles**
- 1

**Strword table**
- ADDR-TOK, ADDR-TOK2

**STRANON**
- ADDR2-MANON

**CMAP stcode**
- CMAP
  1. The postal code can take two optional arguments for the min and max length parameters. Both lengths are defaulted to 6.
  2. The postal code is scanned for non-alphanumeric characters and these characters are filtered out.
  3. If the length is less than Min, the value is treated as ANON. If the length is greater than Max, the value is truncated to the Max value.
  4. The multi-token anonymous values are checked. The mpi_stranon table is checked for a stcode of ADDR2-MANON. The standardized value of the entire address string is compared to the ADDR2-MANON table and, if found, the entire address is marked as ANON.

**CNZIP**
The CNZIP function formats Canadian postal codes.

**Output Type**
- alphanumeric

**Fldargs**
- zipcode

**MinFldArgs, MaxFldArgs**
- 1, 1

**Number of standard roles**
- 1

**MinLength, MaxLength**
- 6, 6

**CMAP stcode**
- CMAP

This function follows the same steps as USZIP (USZIP (standardization function)) on page 262 with the following exceptions/additions:
1. Instead of removing all the digits, the non-alphanumeric characters are removed.
2. The minimum length and maximum length are 6 and 6.

Like USZIP, if the EQUI table is specified and if the attribute does not belong there, it is treated as ANON.

**EMAIL**
This standardization function formats email addresses.

**Output Type**
- alphanumeric with special characters

**Fldargs**
- attrval

**MinFldArgs, MaxFldArgs**
- 1,1

**Number of standard roles**
- 1

**MinLength, MaxLength**
- 0,0

**CMAP strcode**
- CMAP

1. If there is a CMAP table specified, the CMAP character conversion is done first.
2. If there is no @ present in the email, or there is no dot after the @, the attribute is treated as ANON. The value is truncated at the last dot.
3. This function parses the input:
   a. All spaces in the original data are removed.
   b. All underscores ("_") in the original data are converted to a space.
   c. All alphanumeric characters are converted to uppercase.
   d. All other non-alphanumeric characters in the original data are removed.
4. If there is an ANON table specified for this attribute in mpi_dvdstd, the corresponding ANON value table is checked. If this attribute value is found in the ANON table, then it is considered anonymous.
5. If there is an EQUI table specified in mpi_dvdstd, then it is checked for this attribute. If the attribute is not found in the EQUI table, then it is considered ANON.

**Example:**
- aBc.xy34@yahoo.co.uk is converted to ABCXY34YAHOO.CO.

**INTADDR2**
INTADDR2 is a general standardization function for international addresses, with configurable minimum and maximum length parameters for postal codes. A dvdarg of ALLOWHASH can be used to prevent any # (number sign) in the address from being treated as punctuation.

**Output Type**
- any

**Fldargs**
- stline1,stline2,stline3,stline4[city,state,zipcode]

**MinFldArgs, MaxFldArgs**
- 1,7
dvdargs
MinLength, MaxLength, ALLOWHASH

Strword
ADDR-TOK, ADDR-TOK2

Stranon
ADDR2-MANON

The INTADDR2 function follows the same steps as USADDR2 for the street line and region standardization. The difference is in the ZIP code standardization, which for INTADDR2 is as follows.
1. Min and maxLengths are configurable parameters that can be passed as the first two parameters in the dvdargs property of mpi_dvdxstd.
2. If the min and max values are not passed, the default values of 4 (min) and 5 (max) are taken.
3. If the attribute is less than the min, it is treated as ANON. If it is greater than max, it is truncated to the specified max size.
4. If there is an ANON table specified, the attribute is checked against this table and, if found, is marked as ANON.
5. The tokens are then classified according to their type.
6. Finally, the multi-token anonymous values are checked. The mpi_stranon table is checked for a strcode of ADDR2-MANON. The standardized value of the entire address string is compared to the ADDR2-MANON table and, if found, the entire address is marked as ANON.

INTADDR2R
INTADDR2R is a general standardization function for international addresses, with configurable patterns for postal codes. A dvdarg of ALLOWHASH can be used to prevent any # (number sign) in the address from being treated as punctuation.

Output Type
any

Fldargs
stline1,stline2,stline3,stline4[city,state,zipcode]

MinFldArgs, MaxFldArgs
1,7

dvdargs
PATTERNS=PATTERNSET, ALLOWHASH

Strword
ADDR-TOK, ADDR-TOK2

Stranon
ADDR2-MANON

The INTADDR2R function follows the same steps as USADDR2 for the street line and region standardization. The difference is in the ZIP code standardization, which for INTADDR2R is as follows.
1. Valid patterns for postal code are defined as PATTERNS=PATTERNSET in the dvdArgs. The dvdarg entry is not the actual patterns themselves, but the name (strcode) of the patterns.
2. Patterns with PATTERNSET strcode are defined in the mpi_strset table. They specify the correct format for a postal code.
3. If there is an ANON table specified, the postal code is checked against this table and, if found, is marked as ANON.
4. The address tokens are then classified according to their type.
5. Finally, the multi-token anonymous values are checked. The mpi_stranon table is checked for a strcode of ADDR2-MANON. The standardized value of the entire address string is compared to the ADDR2-MANON table and, if found, the entire address is marked as ANON.

**INTZIP**
INTZIP is used to standardize an international postal or ZIP code.

**Output Type**
any

**Fldargs**
zipcode

**MinFldArgs, MaxFldArgs**
1,1

**Number of standard roles**
1

**Dvdargs**

1. The min and maxLengths are taken as optional arguments. They can be passed as the first two arguments of the dvdargs in mpi_dvxstd.
2. If the min and max arguments are not present, the default min (4) and max (5) are used.
3. The non-alpha characters are first removed. If the ZIP code is less than the min length, it is treated as ANON. If the length is greater than max, it is truncated to the max value.
4. If there is an ANON table specified, the attribute is checked against this table and, if found, is marked ANON.
5. If the EQUI table is specified and if the attribute does not belong there, it is treated as ANON.
6. The value is then added to the cmpd string.

**NAADDR2**
The NAADDR2 function standardizes North American addresses.

This function follows the same steps as USADDR2 ("USADDR2" on page 261), except for ZIP code handling. A dvdarg of ALLOWHASH can be used to prevent any # (number sign) in the address from being treated as punctuation.

**Output Type**
any

**Fldargs**
stline1, stline2, stline3, stline4 [city, state, postalcode]

**MinFldArgs, MaxFldArgs**
1,7

**Dvdargs**
ALLOWHASH

**Number of standard roles**
1
Strword table
   ADDR-TOK, ADDR-TOK2

Stranon
   ADDR2-MANON

The ZIP code standardization is as follows:
1. If the number of input values is not equal to 7, an error is issued and no
   further processing is done.
2. The non-alphanumeric characters are removed.
3. If the length is less than 5, the ZIP code is considered ANON.
4. If the length is greater than or equal to 5 and if the first five digits are
   numbers, it is a U.S. ZIP code and it is truncated at the first five digits.
   Otherwise it is considered Canadian postal code and is truncated at the first six
   digits.
5. The token and the type are then added to the address string to form the cmpd
   string. A numeric is preceded by ‘N’ and a string is preceded by ‘S’.
6. After the ZIP code processing, the last part of the address standardization
   handles the multi-token ANON processing similar to the process of USADDR2.
7. The multi-token anonymous values are then checked. The mpi_stranon table is
   checked for a strcode of ADDR2-MANON. The standardized value of the entire
   address string is compared to the ADDR2-MANON table and, if found, the
   entire address is marked as ANON.

NAZIP
NAZIP is used for North American ZIP code standardization (Canada and United
States).

Output Type
   alphanumeric

Fldargs
   zipcode

MinFldArgs, MaxFldArgs
   1, 1

Number of standard roles
   1
1. The non-alphanumeric characters are removed.
2. If the length is less than 5, then the ZIP code is considered ANON.
3. If the length is greater than or equal to 5 and if the first five digits are
   numbers, it is considered a U.S. ZIP code. U.S. ZIP codes are truncated at the
   first five digits.
4. The token and the type are then added to the address string to form the cmpd
   string. A numeric is preceded by an ‘N’ and a string is preceded by an ‘S’.

RZIP
The RZIP standardization function is used to standardize international ZIP (postal)
codes.

Output Type
   any

Fldargs
   zipcode
MinFldArgs, MaxFldArgs
1,1

Number of standard roles
1

dvdargs
PATTERNS=PATTERNSET

Tables mpi_strhead, mpi_strset
1. Non-alphanumeric characters are removed (for example, '-' or ':').
2. Postal code patterns are input by using the derivation arguments and specifying a PATTERNS=PATTERNSET.
3. Patterns with PATTERNS strcode are defined in mpi_strset table. They specify the correct format for a postal code.
   a. If an incoming postal code value does not match any of the defined patterns, it is treated as anonymous.
   b. A pattern can be defined as a combination of 'A's - implying alphabets, 'N's - implying numbers or 'E's implying either. For valid results, patterns are expected to contain a combination of only 'A's, 'E's or 'N's. For instance, in the USA valid patterns are NNNNN or NNNNNNNNNN. This matches with postal codes like 78746 or 78746-1070 (the '-' is removed). For Australia, the pattern is NNNN.

For example, an input postal code of NL-1000 with patterns defined in mpi_strset as EENNNN, NNNN. Then NL1000 ('-' removed) matches with EENNNN (E stands for either number or alphabet).

UKADDR2
This function standardizes United Kingdom addresses.

A dvdarg of ALLOWHASH can be used to prevent any # (number sign) in the address from being treated as punctuation.

Output Type
any type

Fldargs
stline1,stline2,stline3,stline4[city,state,zipcode]

MinFldArgs, MaxFldArgs
1,7
dvdargs
ALLOWHASH

Number of standard roles
1

CMAP strcode
CMAP

The first part of the address standardization is the same as USADDR2. If the number of inputs is greater than 6, then the ZIP code is handled as follows.
1. If the input count is greater than 6, the ZIP code is standardized.
2. The non-alphanumeric characters are first removed and the characters are converted to uppercase.
3. If the length of the ZIP code is less than the Min value (default = 5), the value is treated as ANON.
4. If the length is greater than the Max value (value = 7), the length is truncated to maximum digits.
5. The ANON table is checked and then the token and the token type are added to the address string.
6. After the ZIP code processing, the last part of the address standardization handles the multi-token ANON processing similar to USADDR2.
7. The multi-token anonymous values are then checked. The mpi_stranon table is checked for a strcode of ADDR2-MANON. The standardized value of the entire address string is compared to the ADDR2-MANON table and, if found, the entire address is marked as ANON.

UKZIP
This function is used to standardize UK ZIP codes, which contain alphanumeric characters.

Output Type
any

Fldargs
zipcode

MinFldArgs, MaxFldArgs
1,1

Minlength, Max Length
5,7

1. The non-alphanumeric characters are first removed from the string.
2. The length of the attribute is checked.
   a. If the length is 0, the attribute is treated as ANON.
   b. If the length is less than the Min value (5), the attribute is treated as ANON.
   c. If the length is greater than Max value (7), the attribute is truncated to the max value.
3. If there is an ANON table and if the attributes are listed in this table, it is treated as ANON.
4. If the EQUI table is specified and if the attribute does not belong there, it is treated as ANON.
5. Finally, the output is added to the comparison string.

USADDR
This function is used to standardize U.S. addresses.

Output Type
any type

Fldargs
stline1,stline2,stline3,stline4

MinFldArgs, MaxFldArgs
1,4

Number of standard roles
1

Strword table
ADDR-TOK

CMAP strcode
CMAP table
For most addresses, USADDR produces a string with street number, an optional
directional token, a street name, and a unit number which might be an apartment,
suite, or floor number. For example, 345 NORTH ELM STREET, APT 66 becomes
345_N_ELM_66.

One exception to the formatting is Post Office boxes which are formatted as
POBOX_##.

The algorithm that produces this result is described here.
1. The ADDR-TOK table is loaded by looking up the mpi_strhead and
   mpi_strword tables.
2. The leading spaces are trimmed and a space is added between each line of the
   address input.
3. If there is a CMAP table specified, the CMAP character conversion is done.
4. Uppercase, digit, or space characters are not changed. If there is a lowercase
   character, it is converted to uppercase. Punctuation characters are replaced by a
   space. If there is a # (number sign), there is a space added before and after the
   #. Any other character is taken out and the tokens are collapsed.
5. The address is then parsed into words and the words are then classified
   according to the word type. The maximum number of words possible is 4. The
   classifications are found in mpi_strword under the ADDR-TOK labels.
6. Depending on the token type, the roles are assigned to the address inputs. A
   list of possible token types includes:
   - MPI_TOKTYPE_BN = block number type
   - MPI_TOKTYPE_DT = direction type
   - MPI_TOKTYPE_ST = street type
   - MPI_TOKTYPE_MT = pobox type
   - MPI_TOKTYPE_UT = unit number type
   - MPI_TOKTYPE_SP = special (_MT/_UT)

   Token roles are such that:
   - The box number is assigned when there is a non-MT/SP token
   - Unit number is assigned when there is a non-UT/SP token
   - Block number is assigned when there is a BN token
   - Block number and the actual number are assigned when there is a NUM
     token
   - Direction or street name is assigned when there is a DT token (for
     example, North, N, NE)
   - Street name is assigned when there is an ANT token
   - If the type is ST and the street name is null, then the street type is
     assigned (for example, 123 Circle Court)
   - If the token type is UT or SP, the role is assigned to U (which is Unit)
   - If the token type is MT, the role is assigned to B (which is block)
   - If the token type is NXM and if the previous character was not
     ST,ND,RD,TH, then the street name is assigned. If not, the unit number is
     assigned. For example, 32ND AVENUE assigns 32 to the street name, but
     32 AVENUE ROAD assigns 32 to the unit number.
7. If the box number is null and the street name is null, but there is a direction,
   then the direction is assigned to this street name. If there is a street name, but
   no other sub components (for example, block number, unit number, and so on),
then the street name is considered unsuitable and is given a NULL value. The box number (POBOX) has the highest priority among the token types.

8. Finally the address string is formatted. If there is a box number, the POBOX is added in front of it and the number is appended. Then the street name, street direction, block number, and unit numbers are added, each of them by looking at mpi_strword and getting the equivalent substitution. For example Street is replaced with ST, North is replaced with N, and so on.

USADDR example:

354 North Elm Street becomes 354_N_ELM_ST

**USADDR2**

This function produces a tokenized address string.

The function uses all the tokens, unlike USADDR, and it does not attempt to parse the address string and extract specific tokens. A dvdarg of ALLOWHASH can be used to prevent any # (number sign) in the address from being treated as punctuation.

**Output Type**

Any type

**Fldargs**

stline1,stline2,stline3,stline4[city,state,zipcode]

**MinFldArgs, MaxFldArgs**

1,7

**dvdargs**

ALLOWHASH

**Number of standard roles**

1

**Strword table**

ADDR-TOK, ADDR-TOK2

**Stranon table**

ADDR2-MANON

**CMAP strcode**

CMAP table

1. The function can take in ZIP code and two additional arguments (state and city). These two arguments are used only in comparison when ZIP code is not available.

2. If there are more than six input values, the ZIP code is standardized by using a postal code standardization function. The ZIP code standardization removes all non-digits. The length is verified to be five digits. If the length is less than five digits, the attribute is treated an ANON. If the length is greater than 5, it is truncated to five digits.

3. If there are more than four input values, city and State are standardized by converting into uppercase and removing non-alphabet characters and spaces.

4. If there is at least one input value, the street line is standardized as follows.

a. After trimming any leading spaces, the street lines are separated with a space. The CMAP character conversion is then applied and any punctuation is converted to spaces.
b. The alphabetic characters are converted to uppercase, digits, and spaces are added as they are, and the pound sign (#) is replaced with a space. All other characters are eliminated.

c. A list of tokens is formed by using space as the separator.

d. These tokens are then checked for their abbreviated replacements in mpi_strword and are replaced (for example, EAST becomes E).

e. Multi-tokens are concatenated to form single tokens (for example, N E becomes NE).

f. Each token is then identified as numeric or non-numeric. Numeric tokens contain digits but do not have 1st, 2nd, and so on. An “N” is added to each numeric token and an “S” to each non-numeric token.

5. The multi-token anonymous values are checked. The standardized value of the entire address string is compared to what is in the ADDR-MANON table and, if found, the entire address is marked as ANON.

USADDR2 example:

6045 Elm Street, Cave Creek, AZ, 85331 becomes N-6045:S-ELM:S-STREET:S-CAVECREEK:S-AZ:.N-85331:

ADDR2-MANON table example:

N-123:S-AVENUE:S-AUSTIN:S-TX:N-23232

USZIP (standardization function)
This function formats U.S. ZIP codes.

Output Type
numeric

Fldargs
zipcode

MinFldArgs, MaxFldArgs
1,1

Number of standard roles
1

MinLength, MaxLength
5,5

1. This function first does the CMAP standardization. The non-digits are then stripped from the string.

2. If the length is less than 5, the value is treated as anonymous. If it is greater than 5 it is truncated to 5.

3. The mpi_dvdxstd table is then checked to see if there is an ANON table specified. If specified, the attribute is checked with the table and if the value is found, the attribute is treated as ANON.

4. The mpi_dvdxstd table is then checked to see if there is any EQUI table specified. If specified, the attribute value is checked and if not found, the value is treated as ANON.

5. The attribute value is then added to the comparison string.

USZIP example:
78749-2000 gets converted to 78749
78749A gets converted to 78749
787-49 gets converted to 78749
7874 is treated as ANON

UCSFREQADDR
UCSFREQADDR standardizes addresses that contain characters (for example, Japanese and Chinese) or a combination of characters and Latin. (UCSFREQ stands for universal character set standardization based on universal character frequency counts.)

This function is designed for use only with the AXP comparison function.

When configuring this function, there are two arguments - one required and one optional - defined in the dvdargs column in mpi_dvxstd. The first argument (required) is for country and must be China, Japan, or Korea. The second argument (optional) should be a number that specifies the number of tokens to output. The default for output is 4. The country and output arguments are separated by a comma, for example: China, 4.

The Master Data Engine reads two files per language specified during initialization, either:
• ChinaRanges.txt and ChinaFreq.txt,
• JapanRanges.txt and JapanFreq.txt, or
• KoreaRanges.txt and KoreaFreq.txt

The first file defines the character ranges for that country and the second defines the character frequency (note that the frequency process is mainly used by UCSFREQXNM function.\textsuperscript{[UCSFREQXNM” on page 283].} The ranges are used to tokenize while the Freq is used for defining the bucket tokens.

Note: This function replaces the previously released FORGNADDR and JAPADDR2 functions. The upgrade process handles this name change. Customers that use either of those functions will see the name change automatically in their algorithm configuration.

Output Type
string

Fldargs
Stline1,Stline2,Stline3,Stline4, city, state, postal code

MinFldArgs, MaxFldArgs
7,7

Number of standard roles
1

Other files
Freq.txt and range.txt files
1. This function also takes seven fldargs, even if some of them are empty. The first four are assumed to be the street address and are concatenated with spaces. If there are more than four arguments, then arguments five and six, if they exist, are concatenated and treated as the city and region. If there is a seventh argument, it is assumed to be a postal code.
2. The street address and city and state are tokenized. During tokenization, the input character type is checked to see if it is one of the following: separator, ideograph, punctuation, control, or digits.
   a. If the character type is not known, it is treated as the previous character type.
   b. If it is a separator, punctuation or control character, it is replaced with a space.
   c. If the character type is ideograph, a space is added and the character is written.
   d. All lowercase characters are converted to uppercase.
3. The ZIP code is standardized in the same way as INTZIP ("INTZIP" on page 256).

Attribute standardization functions

Attribute standardization functions are used to format attributes.

There are four attribute standardization functions: ATTR, ATTRA, ATTRN, and MULTIDIM. The ATTR, ATTRA, and ATTRN all use the same processing steps and differ only in the type of attributes (alphabetic or numeric) they standardize.

**ATTR (standardization function)**

This function standardizes any alphanumeric attributes.

**Output Type**

any

**Fldargs**

attrval

**MinFldArgs, MaxFldArgs**

1,1

**Number of standard roles**

1

**MinLength, MaxLength**

0,0

**CMAP strcode**

CMAP table

1. If the input string is not empty and if a CMAP (character mapping) standardization table has been specified, the CMAP conversion is done.
2. All non-alphanumeric characters are filtered out and the alphanumeric characters are converted to uppercase.
3. If the attribute is empty, the attribute is considered ANON since the minimum and maximum lengths are 0. The attribute is not restricted in this case.
4. If the entry in mpi_dvdstd specifies an ANON table in the anonstrcode, the attribute value is checked in the ANON table. If found, then the attribute is considered ANON.
5. If the entry in mpi_dvdstd specifies an EQUI table in the equistrcode, then the attribute value is checked in this table. If not found, the attribute is considered ANON.

**ATTRA**

This function standardizes alphabetic data.
The ATTRA function works in the same way as ATTR except that ATTRA filters out non-numeric characters. ATTRA, as opposed to ATTR, is sometimes used in cases where the characters that are filtered out are not valid (for example, a product code that is only letters).

**Output Type**
- alphabetic (ATTRA)

**Fldargs**
- attrval

**MinFldArgs, MaxFldArgs**
- 1,1

**Number of standard roles**
- 1

**CMAP strcode**
- CMAP table

**ATTRN**

ATTRN standardizes numeric attribute data.

The ATTRN function works in the same way as ATTR except that ATTRN filters out non-alphabetic characters and converts them to uppercase. ATTRN, as opposed to ATTR, is sometimes used in cases where the characters that are filtered out are not valid (for example, a serial number that must be purely numeric).

**Output Type**
- numeric (ATTRN)

**Fldargs**
- attrval

**MinFldArgs, MaxFldArgs**
- 1,1

**Number of standard roles**
- 1

**MULTIDIM**

This function standardizes attributes that have multiple fields. It is intended for use with the DR1D[234][ABC] comparison functions.

For this function, every field argument in the attribute is standardized as a separate dimension. For example, you have an identifier attribute with two fields that are primaryID and secondaryID. The primaryID is standardized in one dimension and secondaryID is standardized in another dimension. During comparison, the primaryID compares only to the first dimension and secondaryID compares to only the second.

**Fldargs**
- From 2 to 4 fields. The first field specified is dimension 1, the second is dimension 2, and so on.

**MinFldArgs, MaxFldArgs**
- 2,4

**Number of standard roles**
- 1

**MinLength**
- 0
MaxLength

Unrestricted but the resultant standardized value must fit in 381 bytes when encoded as UTF8.

CMAP strcode

CMAP table

Each field is standardized into a separate dimension according to the following rules:
1. If a CMAP is specified, each character is translated according to the character mapping specification.
2. Any character that is not alphanumeric is removed.
3. Alphabetic characters are translated to uppercase.
4. If an ANON value set was specified and if the standardized value matches one of the ANON values, the standardized value is a zero-length string.

A standardized multidim value looks like: 12345+56789. The ‘+’ indicates that they are different dimensions within the same role.

Biometric standardization functions

Biometric functions are used to standardize items like hair or eye color, race, height, and weight.

There are standardization functions that handle various biological attributes.

EYECOLOR, HAIRCOLOR, RACE

These three functions behave in a similar manner to the ATTR function to format eye color, hair color, or race as alphanumeric data.

All non-alphanumeric characters are filtered out. An EQUI table is used to allow only valid inputs.

Output Type

alphanumeric

Fldargs

attrval

Number of standard roles

1

CMAP strcode

CMAP table

1. If the input string is not empty and if a CMAP standardization table has been specified, the CMAP conversion is done.
2. All non-alphanumeric characters are filtered out and the alphanumeric characters are converted to uppercase.
3. If the attribute is empty, the attribute is considered ANON since the minimum and maximum lengths are 0. The attribute is not restricted in this case.
4. If the entry in mpi_dvdxstd specifies an ANON table in the anonstrcode, the attribute value is checked in the ANON table. If the attribute is empty, it is considered ANON.
5. If the entry in mpi_dvdxstd specifies an EQUI table in the equistrcode, then the attribute value is checked in this table. If not found, the attribute is considered ANON.
HAIRCOLOR:
BROWN is converted to BR
BLONDE is converted to BL

EYECOLOR:
BROWN is converted to BR
BLUE is converted to BL

HEIGHT
The HEIGHT function converts the string to a numeric value.

Output Type
numeric (feet and inches)

Fldargs
attrval

Number of standard roles
1

MinValue, MaxValue
36,90
1. If conversion to a numeric value is not possible, then the attribute is treated as ANON.
2. It is assumed that the input value is a three-digit integer, with the first digit being feet and the last two digits being inches (FII). This numeric value (which is in feet and inches) is converted to inches. For example 511 is equal to 5 feet, 11 inches and is ultimately converted to 71.
3. This value is then checked to see if it falls between 36 and 90. If it does not fall within that range, the attribute is treated as ANON.
4. The attribute is then converted back to a string format and added to the comparison string.

A height of 5'0 is converted to 60 inches.

WEIGHT
This function standardizes physical weight.

Output Type
numeric

Fldargs
attrval

Number of standard roles
1

MinValue, MaxValue
60,500
1. The weight is stored in pounds and is standardized using this function.
2. If the weight is not numeric, it is treated as ANON value.
3. If the weight is less than 60 or greater than 500, the value is treated as ANON.
4. The final value is added to the comparison string.
**Date standardization functions**

Date functions standardize date strings.

The strings can be dates, incomplete or approximate dates, and ages.

**AGE**

This function standardizes ages.

**Output Type**

numeric

**Fldargs**

attrval

**dvdargs**

C (circa option)

**Number of standard roles**

1

**MinValue, MaxValue**

8,100

1. This function first converts the attribute value from string to numeric. If this conversion is not successful, the attribute is treated as ANON.

2. If the age is less than 8 or greater than 100, it is treated as ANON value.

3. The age values are then converted to birth year by subtracting the value from the current year value. If the current month is less than 7, the year value is reduced by 1. The assumption is in this case is that the birthday is yet to come.

**Note:** This function has a circa option which can be used when the age is an estimated age. You can pass a dvdarg of 'C' and a "C" prefix is added to the standardized data.

**AGE example:**

A value of 30 is converted to 2006-30=1976. If the current month was March (<7), then 30 is converted to 2006-30=1976 -1=1975.

**CDATE**

The CDATE standardization function is part of the algorithm solution used to compare incomplete, partially valid or approximate date, or partially valid and approximate date ranges. The solution includes the CDATE standardization, bucketing, generation, and comparison functions.

The date range is specified by a resolution parameter given by the number of days, months or years prefixed by D, M, or Y. For instance given a resolution value of M4, the date range is then four months before and after the given date.

**Output Type**

numeric (date)

**Fldargs**

dateval, resolution

**dvdargs**

min year, max year, resolution

**MinFldArgs, MaxFldArgs**

1, 2
Number of standard roles
1

1. The input to the function comes from an implementation-defined segment (for example, mpi_idsdate, with input fields defined as varchar.

2. The resolution parameter can be specified as part of the dvdargs (a single resolution for all the dates) or it can be specified individually for each input date. To specify individually for each input dateval, you must specify the input with two fields - dateval and resolution. Then you specify these two fields in the fldargs parameter in the properties settings in IBM Initiate Workbench for CDATE. All valid resolutions must be prefixed by a Y, M or D. Any other prefix is considered invalid and the date/resolution for the member is treated as an anonymous value. If a resolution value is not specified in the dvdargs, implying an exact date, the default resolution value of D0 is used.

3. The date parts (day, month, and year) are stored separately, together with the resolution parameter to get a four-part comparison role. The dateval part of the standardized string is standardized much like the DATE2 standardization function (performing month/day validations, handling missing date parts, anonymous checks, and so on). The only difference from DATE2 is that if a transposed date is used in the output, it is not with a prefixed ‘T’.

4. The standardized string from CDATE is of the form YYYY.MM.DD.RESOLUTION

For example, if the Date were 10.30.2009 and resolution specified as D36 (implying 36 days), then the standardized string would be 2009.10.30.D36.

DATE1
This function formats alphanumeric date data.

Output Type
numeric

Fldargs
dateval

MinFldArgs, MaxFldArgs
1,1

Number of standard roles
1

Stranon table
DATE

1. This function first takes the input and extracts all the digits. All alphabetic characters are removed.

2. The length of this date value is then checked. If the length is 0, the value is treated as ANON. If the length is greater than 8, then the value is truncated to the first eight digits.

3. The value is then checked to see that it does not convert to an invalid date (for example, if the year is less than 1 or greater than 9999, if the month is not between 1 and 12, and so on).

4. If step 3 fails, the attribute is treated as ANON.

5. If there is an ANON table specified in mpi_dvdxstd, then the corresponding table is checked for this attribute value. If found in the ANON table, the attribute is treated as ANON.

A date value of 1971-23-01 or 23-01-1971 or 01/23/1971 is converted to 19710123.
DATE2 (standardization function)

Using the DATE2 standardization function along with the DATE2 comparison function enables comparison of incomplete or partially valid dates.

Because incomplete or partially valid dates are removed when stored in the mpi_memdate segment, you might want to store these dates in mpi_memattr.

**Output Type**

Numeric (date)

**Fldargs**

dateval or attrval

**Dvargs**

Min year, Max year

**MinFldArgs, MaxFldArgs**

1,1

**Number of standard roles**

1

1. This function outputs an eight-digit string. If year, month or day is invalid, those portions of the date are replaced by a string of 0's (zeros) and the comparison function ignores the 0 string.

2. The function is configurable with a minimum and maximum year. When the date falls outside of those ranges, the entire date is treated as an anonymous value.

3. After applying the year range filter, DATE2 checks the date against a configurable date standardization table.

4. The entries for this table are stored in mpi_strequi table that has length 8 and length 4 entries. The length 8 entries are DATE and length 4 entries are MONTHDAY. DATE entries map the entire input string to a standard string. The MONTHDAY string maps only the month and day portions of string. This means that strval2 is replaced by strval1.

<table>
<thead>
<tr>
<th>STRVAL2</th>
<th>STRVAL1</th>
<th>WORDTYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0101</td>
<td>0000</td>
<td>MONTHDAY</td>
</tr>
<tr>
<td>20070229</td>
<td>20070301</td>
<td>DATE</td>
</tr>
</tbody>
</table>

In the example:

1. The first entry specifies that any month and day of 0101 should be mapped to 0000 or marked as invalid. Since the year is left as is, for any date with a month and day of January 1, only the year is used.

2. The second entry specifies that the date 20070229 should be changed to 20070301.

3. After applying the date standardization table, the month and date is checked. If the month is between 1 and 12, the month is left as is. Otherwise, the month is set at 00.

4. If the output month is not valid and the day is between 1 and 31, the day is left as is. Otherwise, the day is set to 00.

5. If the month and year are valid, the month and year are used to determine if the day is valid. If the day is valid, it is left as is. Otherwise, the day is set to 00.
6. If the month is valid, but the year is not, then determination is made on whether the day is valid (leap year is assumed). If the day is valid, it is left as is. Otherwise, the day is set to 00.

7. After completing the day analysis, if one or both of the month and day entries is invalid:
   a. Return to the original month and day and transpose month/day. Analysis is repeated.
   b. If analysis yields a valid month and day, the transposed date is used in the output with a prefix of 'T'.

**GRDATE**

This function formats the date in the same way as DATE1, but eliminates dates where the year is less than 1890 or greater than 2020.

**Output Type**

alphanumeric

**Fldargs**

dateval

**MinFldArgs, MaxFldArgs**

1,1

**Number of standard roles**

1

**Stranon table**

DATE

1. This function follows the first three steps of the DATE1 process.
2. If the date conversion did take place, but the year, month or dates had zero values, the attribute is treated as ANON.
3. The date year is then checked to see if it is less than 1890 or greater than 2020. If either of these conditions is true, the date value is treated as ANON.
4. The mpi_dvdxstd table is checked to see if any ANON table is specified. If it is, then the date value is checked to see if it is found in the table. If the value is found, the attribute is treated as ANON.
5. The value is then added to the comparison string.

**Geographic standardization functions**

**GEO (standardization function)**

The GEO standardization function converts latitude/longitude location coordinates into a standardized format that can be consumed by GEO comparison and bucket functions.

The mpi_memaddr and mpi_memelig tables each contain a 40 character-length geotext1field that stores the latitude/longitude. If you are creating an implementation-defined segment (IDS), your IDS table requires a 40 character field for latitude/longitude storage. The data used to derive the latitude and longitude should be in the same row as the associated geotext field. Since the latitude/longitude is typically derived from the member address, memaddr is suggested.

**Output Type**

alphanumeric
1. The input format accepted by GEO consists of a latitude coordinate followed by a longitude coordinate. The latitude is separated from the longitude by one or more spaces or tab characters. The latitude/longitude coordinates must be valid (-90 <= lat <= +90, -180 < long <= +180) or the geocode is treated as a missing or anonymous value.

2. Both the latitude and longitude consist of a sign and an angle.
   a. For a latitude, the sign is a single character from the set “+NS”. The characters “+” and “N” denote a north latitude and the characters “-” and “S” denote a south latitude.
   b. For a longitude, the sign is a single character from the set “+EW”. The characters “+” and “E” denote an east longitude and the characters “-” and “W” denote a west longitude.
   c. The sign for latitude and longitude might be omitted. If omitted, the sign of the latitude is assumed to be “N” and the sign of the longitude is assumed to be “E”.
   d. For both latitude and longitude, the angle might be specified in either degrees/minutes/seconds/fractional seconds or degrees/fractional degrees. Minutes, seconds, and fractions are separated from each other by a period (.). In order to avoid ambiguity, when specifying degrees/minutes/seconds all three values must appear.

   For example, all of the following inputs represent the coordinate with latitude 30 degrees, 17 minutes, 59 seconds north, and longitude 97 degrees, 41 minutes, 59 seconds west:
   - N30.17.59 W97.41.59
   - +30.17.59 -97.41.59
   - 30.17.59 -97.41.59

   To express the same coordinate in degrees and fractions, the following forms can be used:
   - N30.3 W97.7
   - +30.3 -97.7
   - 30.3 -97.7

3. The standardized form represents both latitude and longitude as degrees/fractional degrees. Latitude is separated from longitude by the character ‘X’. The letters “NSEW” are used to represent the sign of the angles and the decimal point is represented by the character “V”.

The standardized form of the coordinate described above is:

N30V300000000000XW97V700000000000
The default precision of the standardized coordinates is 12 decimal places. This precision can be changed by using an optional argument to the standardization function. Specify the precision by entering the number XX where XX is the number of decimal places of precision wanted.

**Identifier standardization functions**

Identifier standardization functions offer various standardization methods for the input formats and types of ID numbers passed to the Master Data Engine.

**IDENT1**

IDENT1 is used to standardize alphanumeric data associated with ID numbers.

All special characters are filtered out. This function is used often for standardizing Medical Record Numbers (MRNs).

**Output Type**

alphanumeric

**Fldargs**

idnumber

**MinFldArgs, MaxFldArgs**

1,1

**Number of standard roles**

1

**Stranon table**

SSA (in the case of SSN)

1. The ID number input data is not associated with an issuer source during standardization.
2. The alphanumeric characters in the attribute are extracted and converted to uppercase.
3. The minimum length and maximum length values are read from mpi_srchead. If the attribute length is not between the minimum length and the maximum length, it is treated as ANON. If the minimum length and maximum length values are 0, no check is performed.
4. Since the IDENT functions use only the data in mpi_memident and each row in mpi_memident has an associated memrecno, each attribute is associated with a unique mpi_srchead row. If there is an ANON table whose strcode matches that source code, this attribute is marked as ANON.
5. If there is an ANON table specified in mpi_dvdxstd, that table is checked.
6. The attribute is then added to the comparison string.

675-asd-7008 is converted to 675ASD7008

902-55-5432 is converted to 902555432

**IDENT1A**

IDENT1A, like IDENT1, is used to standardize alphanumeric data associated with ID numbers, but filters out all non-alphabetic characters.

IDENT1A follows the same steps as IDENT1, but filters out all numbers and special characters.
IDENT1N
IDENT1N, like IDENT1, is used to standardize alphanumeric data associated with ID numbers, but filters out non-digit characters.

IDENT1N follows the same steps as IDENT1 (and filters out non-digits characters). This function works well with Social Security numbers.

IDENT2
This function works similarly to IDENT1 in that it formats alphanumeric data, removing special characters. The difference is that this function appends an issuer code (source identifier) to the number.

This function is often used with driver licenses and passports where you must add the state or country to the identifiers.

Output Type
alphanumeric

Fldargs
idnumber

MinFldArgs, MaxFldArgs
1,1

Number of standard roles
1

1. The IDENT2 function adds the issuer source code to the ID number and works in the same way as IDENT1.
2. The srchead value from mpi_srchead, followed by a ‘-‘ (dash) and the ID number, is added to the comparison string.

IDENT2 example:

An SSN value of 666-03-1899 with an issuer source SSA in mpi_srchead would be converted to SSA-666031899.

IDENT2A
IDENT2A follows similar steps as IDENT2 to format alphabetic identifiers by removing numbers and special characters.

Like IDENT2, this function appends an issuer code.

IDENT2N
IDENT2N follows the same steps as IDENT2, but filters out all non-digit characters.

It also appends an issuer code to the number.

Name standardization functions
Standardization functions are provided to handle formatting person and business names.

BXNM (standardization function)
This function is used for business name standardization with specialty codes, and formats a list of strings.
BXNM is the only standardization function that uses two standard roles:

- Role 1 extracts the names and converts them to standardized tokens
- Role 2 extracts the specialty codes and standardizes them

**Output Type**
any value

**Fldargs**
onmlast,[onmfirst, onmmiddle, onmsfx, onmpfx, onmdgr]

**MinFldArgs, MaxFldArgs**
1,6

**Number of standard roles**
2

**Strequi table**
BXNM-ABS

**Strword table**
BXNM-TOK

**Stredit table**
BXNM-SED

**CMAP strcode**
CMAP table

1. This function first loads all the tables associated with BXNM-SED, BXNM-TOK, and BXNM-ABS.
2. If there is a CMAP table specified, the CMAP character conversion is done before standardization.
3. The attribute is converted to all uppercase, any backslash or quotations are removed, and a space is added to the end of the string.
4. The spaces between single characters are deleted and the characters are concatenated. For example, A B C becomes ABC.
5. A lookup in mpi_stredit table is performed with the attribute value as the input string value (inpstrval). This value defines either a single- or multi-token pattern, and the corresponding output string value (outstrval) that defines a replacement string for the attribute value. For both input and output, multiple tokens are separated by a space.
6. The input values are converted to words (including any abstract code). This conversion enables them to be used later to get the frequency counts based on the word type.
7. The word types are retrieved from the mpi_strword table(*) and sorted by frequency. Because there is a maximum word limit of four for BXNM, only the first four words are taken and the rest are discarded. The maxwords property can be changed to a maximum of 16 by passing the value in the dvdargs property.
8. The ABS code part is not included in step 6. The mpi_strequi table is looked up using the ABS code and the equivalent value from the table replaces this value. This value is added to the comparison string if the second standard role is also specified in mpi_dvdxstd.

* Token types are:
  - ANT (alphanumeric)
  - NUM (all numeric)
  - NMX (leading/trailing digits)
• BT (business type)
• PST (person suffix type)
• PCT (person credential type)
• PPT (person prefix type)
• BLT (business legal type)

BXNM example:

GEORGE COSTAN’A J R, DENTIST is converted to COSTANA:GEORGE:JR:DNT

**CJKSTD**

Many organizations that store Chinese, Japanese, or Korean data find that organizational names are not linked because their ideograms contain “company” or “inc” designations.

The CJKSTD standardization function supports the removal of anonymous substrings at the beginning and the end of a token. CJKSTD is designed to handle organization names that include morphemes for “company”, “incorporated” with no token separation.

**Output Type**

alphanumeric with special characters

**Fldargs**

Corporation name (MEMATTR.attrval, MEMNAME.onmlast)

**MinFldArgs, MaxFldArgs**

1,1

**Number of standard roles**

1

**strconfig table**

cfgtype = CJKSTD

**stranon table**

ANON

**Dvdargs**

maxWords=value,cjkStd=cftype (key=value)

**CMAP strcode**

CMAP

The configuration for the CJKSTD function is obtained from the mpi_strconfig table. The cfgtype (configuration type) for the standardization function must be CJKSTD.

1. If there is a CMAP table specified, the CMAP character conversion is done before standardization.

2. The cjkStd property is obtained from the dvdargs. It is a mandatory argument to pass. This argument determines the algorithm configuration to be obtained from the mpi_strconfig table.

3. The maxWords property can be passed in the dvdargs property value. If no value is specified, the default of 4 is taken. If the value is passed, then the lower of the dvdargs value or 16 is used.

4. The algorithm configuration for CJKSTD, which is obtained from mpi_strconfig table, is of the following form: type name low high;sep codepoint;ideograph typename. Where:
• type name - signifies a new character type
• low, high- specifies a range (low and high code points); specifying a high code point is optional
• sep - indicates a separator character
• codepoint - code point of the separator character
• ideograph typename - indicates which character range represents ideographs

For example, the entry in the strVal for CJKSTD configuration looks like:

type cjk 0x4E00 0x9FFF;sep59;ideograph cjk;

5. The next step is tokenization. A space, a separator character (specified in the configuration as sep codepoint) or a different character type (other than the one specified in type name low high) ends a token.

6. If the buffer formed from the above step is null, or if there is an ANON table specified in mpi_dvdxstd and if the input is present in mpi_stranon, then it is treated as ANON.

7. The buffer is then parsed into words. If a word is a string of ideographs, the anonymous substrings are removed from the beginning and end.

8. The words are added until the limit specified by maxWords is reached.

9. Finally the words are checked against the ANON and EQUI values (if specified with mpi_dvdxstd row) and the final cmpd string is obtained.

**CJKCXNM**

This function standardizes company names that use ideographic characters (for example, Chinese, Japanese [Kanzi] and Korean).

**Output Type**

any value

**Fldargs**

onmlast,[onmfirst, onmmiddle, onmsfx, onmpfx, onmdgr]

**MinFldArgs, MaxFldArgs**

1,1

**Number of standard roles**

1

**Dvdargs**

maxwords (a numeric value)

**CMAP strcode**

CMAP

1. The function checks to see if there is any data to be processed in the input buffer.

2. If there is a CMAP table specified, the CMAP character conversion is done before standardization.

3. The maxwords property can be passed in the dvdargs property value. If no value is specified, the default of 4 is taken. If the value is passed, then lower of the dvdargs value or 16 is used.

4. If the preceding character was an ideographic character, then an ideographic character is written to the buffer.

5. If there is punctuation, the punctuation is removed and the space collapsed.

6. If there is a space, verify that there is at most one space written.

7. If the character is in a different Unicode block and it is not ideographic, then characters are added after a space.
8. If there is an alphabet character, it is converted to uppercase (if in lowercase) and then added.
9. If the input is not an ideographic, punctuation, space, code block change, or alphabetic character, then it is added to the buffer and the final space terminated.
10. If the buffer formed from the above steps is null or if there is an ANON table specified in mpi_dvdxstd and if the input is present in mpi_stranon, then it is treated as ANON.
11. The buffer is then parsed into words and the words are added until the limit of four is reached.
12. Finally, the words are checked to see if they are present in mpi_stranon and mpi_strrequi (if specified in mpi_dvdxstd). The appropriate functionality is applied for ANON and EQUI values.
13. The final cmpd string is then formed with this output.

**CXNM (standardization function)**

Used for business name standardization, the CXNM function provides two different kinds of anonymous values: single-token and whole-value.

Single-token anonymous values are removed on a per-word basis whereas whole-value anonymous values cause the entire name to be treated as anonymous (missing). Whole-value anonymous values must be enclosed in parentheses to distinguish them from single-token anonymous values. Single-token anonymous values do not have enclosing parentheses.

**Output Type**
- any value

**Fldargs**
- onmlast

**MinFldArgs, MaxFldArgs**
- 1

**Number of standard roles**
- 1

**Dvdargs**
- maxwords (an integer value)

**CMAP strcode**
- CMAP

1. The function first checks to see if there is any data in the input to be processed.
2. If a CMAP table is specified in the mpi_dvdxstd table, then the CMAP conversion is done first.
3. The lowercase characters are converted to uppercase. Any uppercase characters and digits are left as they are. Any ampersand, single quotation, or back quotation is deleted and the characters are joined (for example, AT&T becomes ATT).
4. Single characters are combined together by deleting the space between them. The final space is then deleted from the buffer (for example, I BM is converted to IBM).
5. If there is an ANON table specified in mpi_dvdxstd, then the input is checked to see if the whole value is anonymous. A value that is enclosed in parenthesis is considered a whole value ANON.
6. The input is then separated into words. The words are added until the word limit is reached. This value can be passed through the dvdargs property. The maximum value is 16. If the dvdargs is not specified, a default of 4 is used.

7. If there is an ANON table specified in mpi_dvdxstd, each word is checked to see if it is anonymous.

8. If the EQUI table is specified in mpi_dvdxstd and if the word is found in mpi_strequi, the word is replaced with the first entry in the EQUI set.

Whole-value example:

If UNKNOWN was specified as anonymous, the name "UNKNOWN" would correctly become anonymous, and the name "WORLD OF THE UNKNOWN" would be standardized as WORLD OF THE. However, if you specify (UNKNOWN) as a whole-value ANON and INC as a single-token ANON, the name "WORLD OF THE UNKNOWN INC" would correctly standardize to WORLD OF THE UNKNOWN. Also the value UNKNOWN would correctly become anonymous.

**PXNM (standardization function)**

This function handles person name standardization.

**Output Type**

any value

**Fldargs**

onmlast,[onmfirst, onmmiddle, onmsfx, onmpfx, onmdgr, onmtitle]

**MinFldArgs, MaxFldArgs**

1,6

**Number of standard roles**

1

**Dvdargs**

maxtokens (a numeric value)

**CMAP strcode**

CMAP

1. The different tokens of the name are extracted from the attribute (first name, last name, middle name, suffix, prefix, and degree) into components.

   **Note:** The field order in which PXNM presumes the tokens is 1) last name, 2) first name, 3) middle name, 4) prefix, 5) suffix, and 6) degree. The behavior of the last three fields is such that only words found in the string tables PXNM-PFX, PXNM-SFX, or PXNM-DGR are standardized. In the first three fields, all words are standardized as long as the word is not found in PXNM-LNJ, PXNM-FNJ and PXNM-MNJ (where “J” = join).

2. The order of token processing is: suffix, prefix, degree, first name, middle name, last name.

   a. These components are first standardized by applying the CMAP conversion if a CMAP table is specified. Single quotes and backslashes are eliminated. All lowercase characters are converted to uppercase and a space is added at the end of the string (if the last character was not a space).

   b. The standardized words or tokens are then checked to see which component they are (suffix, prefix, degree, last name and so on). This is done by checking each token to see if it belongs to the PXNM-SFX, PXNM-PFX or PXNM-DGR tables. The tokens are categorized according to which table they belong.
c. The PXNM-SFX, PXNM-PFX, and PXNM-DGR tables are stored in mpi_strword. The mpi_strword table contains two columns: strval and strstd. The tokens are checked in the PXNM-SFX, PXNM-PFX, and PXNM-DGR tables for a match. For example, if a token matches the strval in PXNM-SFX, it is considered a suffix and is standardized by the strstd function.

d. If first name, middle name and last name do not belong to any of the above tables, they are considered first, middle or last depending on which component is being processed.

e. For last names only, all single character tokens at the end of the string are eliminated. For example, JOHN A B becomes JOHN.

3. If the first, last, or middle name token contains any digits, the entire token is not added to the comparison string.

4. In addition, for the first, last, and middle names, disjointed-word tokens are checked and are concatenated. The PXNM-LNJ, PXNM-FNJ, and PXNM-MNJ tables, which are in mpi_strword, are used in these cases. These tables work in the same way as previously mentioned and are configurable. For example, DE CATHLENIE becomes DECATHLENIE.

5. Anonymous tokens are eliminated.

6. If there are multiple suffix tokens, only the first token is added to the cmpd string. The tokens are added to the cmpd string until the maximum token limit is reached. The max tokens limit is a dvdargs parameter. The maximum value is 16, with a default of 4.

RXNM (standardization function)
The RXNM standardization function is used to split and rewrite (name) tokens. This function allows for standardization of input that contains non-Romanized letters. For tokenization, this function can be used with the built-in rules (RXNMARABICRULES) or with custom tokenization rules. RXNM standardization is designed for use with the QXNM comparison function. Use the PXNM bucketing function with RXNM (EQMETA can also be used to provide nickname phonetics). Any of the bucket generation options are valid. The phonetic function should be ARABICPHONE, unless you have developed a custom phonetic function.

Output type
  Alphanumeric

Fldargs
  MEMNAME, MEMATTR

MinFldArgs, MaxFldArgs
  None

Number of standard roles
  1

MinLength, MaxLength
  None

dvdargs
  STRCODE (reference from mpi_strconfig), NOSKIPTOKEN

mpi_strconfig strcode
  RXNMARABICRULES or user defined

mpi_stranon strcode
  ANON
mpi_strequi strcode
   EQUI

mpi_strcmp strcode
   CMAP

1. The lookups are completed during standardization and transforming the input
token to an alternate form. Unless the alternate form is anonymous, it is used
to create the resulting cmpval stored in mpi_memcmpd. These lookups are
performed on sequences generated from user-defined rules too. Lookups
include:
   • PFX - Prefix lookup table. Defined in mpi_strword with the following strcode
     RXNM-PFX
   • SFX - Suffix lookup table. Defined in mpi_strword with the following strcode
     RXNM-SFX
   • DGR - Degree lookup table. Defined in mpi_strword with the following
     strcode RXNM-DGR

2. Two dvdargs are available: a strcode referencing an encoder type from
   mpi_strconfig (required) and NOSKIPTOKEN (optional).
   a. A strcode reference to a rule set is required to use RXNM. A default specific
to Arabic names called RXNMARABICRULES is provided with the Engine
installation. New rule sets can be added to mpi_strconfig by associating
them with the cfgtype of ENCODER. The strcode indicating the rule set
must be the first argument.
   b. NOSKIPTOKEN is an optional argument is used to persist the original
input token into the final cmpval along with any tokens generated by rule
processing. For example, if the input name is ABDULSATAR and a rule is
defined to generate ABD AL SATAR, the final cmpval would look like:
   ABDULSATAR:ABD:AL:SATAR. Otherwise, by default the cmpval would
look like: ABD:AL:SATAR. In practice, NOSKIPTOKEN should not be used,
but is provided for customers that might require it. You will get better
comparison results if you do not use the NOSKIPTOKEN option.

For information about custom phonetic functions and associated syntax, see
"Custom Phonetics and Rule Sets."

Custom Phonetics and Rule Sets:

Custom phonetic functions are configured using the STRCONFIG facility. A
phonetic function uses the 'ENCODER' cfgtype in mpi_strconfig.

Custom phonetic functions can be defined using the language described below.

Note: Custom encoders can also be used to define the behavior of the RXNM
standardization function. See "RXNM (standardization function)" on page 280 also.

Encoder - Phonetic encoder

An Encoder consists of a sequence of RuleSets and an optional character
remapping.

An Encoder consists of a sequence of RuleSets and an optional character
remapping. To encode a given string,
1. The input is first remapped, if any remapping has been specified for the
   encoder.
2. Each RuleSet is applied in-order.
3. The input of the first RuleSet is the remapped string.
4. The output of the Encoder is the output of the final RuleSet.

The output of each intermediate RuleSet becomes the input to the next RuleSet.

**Syntax**

An encoder is defined as a sequence of rules and character remappings using the syntax described below. Also supported are inline comments enclosed in pound symbols (#). By default, an Encoder contains a single RuleSet that contains all Rules. Multiple RuleSets are separated by the rule set delimiter character - percent symbol (%).

**Remapper**

A Remapper consists of a set of mappings for characters. Each character mapping has the syntax:

'<' replaceChar equivalentChars+ '>'

where replaceChar is the character that will replace the input character, and equivalentChars are the characters that will be replaced.

**Rule set**

A RuleSet defines an encoding of a string. To encode a string, each character of the string is examined. At each character position, a match of a subset of the rules in the set is attempted. The current character determines the subset of the rules to use for matching. If the character is an ASCII letter, the rules for that letter are used. Otherwise the default set of rules is used. The rules are tried in-order until a match is found. After a match, the replacement string of the matching rule is applied and no more rules are tried at that position. Matching continues at the next character after the window (if any).

If no matching rule is found, the character at the current position is deleted. Matching continues at the next character position (if any).

**Rule**

A rule is a multi-character pattern to apply at a given position in a string. If matched, part of the matched text can be replaced.

```
rule
  '@' initialCharacter ruleDelimiterChar charPattern* '(' charPattern+ '=' replacement ')' charPattern* ruleDelimiterChar

initialCharacter
  A single character. Rules are partitioned according to this character. Only ASCII letters will yield a performance improvement.

ruleDelimiterChar
  A single character that is not used as a literal within the rule. This character delimits the end of the rule.

charPattern
  A pattern to match a single character. These are the types of patterns available:
CP_START
  Matches at the beginning of the string.
  Syntax: '^'
CP_END
  Matches at the end of the string.
  Syntax: '$'
CP_ANY
  Matches any character.
  Syntax: '.'
CP_SINGLE
  Matches a single specific character.
  Syntax: character
CP_SET
  Matches any of a given set of characters. If inverted, matches any character not in the given set.
  Syntax: '[' characters... ']' 
  For inverted set: ['!' characters... ']'
CP_SAME
  Matches a character if it is the same as the character at the given relative position.
  Syntax: '{' relativePosition '}'

For CP_SINGLE and CP_SET, characters can be literal or escaped. The escape syntax is \uxxxx where xxxx is the hexadecimal Unicode codepoint value.

replacement
  Any sequence of characters (except ')' or '.'). Can be empty to delete the original characters. Use '.' to indicate no replacement (i.e. preserve original characters). Unicode escapes are also valid here. See the syntax in the charPattern description.

  In this notation, * means zero or more, and + means one or more. The parentheses define the "window" as described below.

A rule consists of a sequence of CharPatterns. The patterns must match in-order for the rule to match. The alignment of the patterns with the string is specified by the window. The first pattern in the window is matched against the current position in the string. Any patterns that precede the window are matched against characters before the current position.

If all patterns in the rule match, the rule itself is considered a match. In this case, a portion of the string can be replaced. The portion of the string that is replaced is specified by the window.

UCSFREQXNM
This function is used to format names with UTF format and is designed for use with the CXNM comparison function. (UCSFREQ stands for universal character set standardization based on universal character frequency counts.)

This function is a generalization of the JAPXNM function from previous releases.
As with UCSFREQADDR (“UCSFREQADDR” on page 263), two arguments - one required and one optional - are defined in the dvdargs column in mpi_dvdxstd. The first argument (required) is for country and must be China, Japan, or Korea. The second argument (optional) is a number that specifies the number of tokens to output. The default for output is 4. The country and output arguments are separated by a comma, for example China, 4.

During initialization the Master Data Engine reads two files per language specified, either:
- ChinaRanges.txt and ChinaFreq.txt,
- JapanRanges.txt and JapanFreq.txt, or
- KoreaRanges.txt and KoreaFreq.txt.

The first file defines the character ranges for that country and the second defines the character frequency. The ranges are used to tokenize while the Freq are used for defining the bucket tokens.

**Note:** This function replaces the previously released FORGNADDR and JAPADDR2 functions. The upgrade process will handle this name change. Customers that use either of those functions see the name change automatically in their algorithm configuration.

**Output Type**
- foreign characters (UTF included)

**Fldargs**
- any

**MinFldArgs, MaxFldArgs**
- 1,4

**Number of standard roles**
- 2

**CMAP strcode**
- CMAP

**Other Files**
- Frequency and range files (for example, Chinese)

1. UCSFREQXNM takes in a maximum of four fldargs as the input. The input can include any characters, including UTF format.
2. If there is a CMAP table specified, the CMAP character conversion is done first.
3. During tokenization, the input character type is checked to see if it is one of the following: separator, ideograph, punctuation, control or digits.
   a. If the character type is not known, it is treated as the previous character type.
   b. If it is a separator, punctuation or control character, it is replaced with a space.
   c. If the character type is ideograph, a space is added and the character is written.
   d. All lowercase characters are converted to uppercase.
4. After the initial processing is done, the input is parsed into words. The max allowable value is 64.
5. The ANON and EQUI tables are checked if specified. If the word is in the ANON table, it is treated as ANON. If the word is in the EQUI table, the corresponding EQUI value replaces the word. If the EQUI table returns more than one value, the first value is taken.

6. The bucket value is then generated by counting the character frequencies and choosing the least three frequency tokens to form the buckets.

7. The bucket role 2 is used in bucketing of the foreign names.

**Phone standardization functions**

These functions standardize phone numbers.

Phone functions are available to handle the standardization of local and international phone numbers.

**AUSTRPH**

AUSTRPH standardizes an Australian phone number to an eight-digit local number.

The phone number might have leading characters which are eliminated by the standardization functions.

**Output Type**

Numeric

**Fldargs**

phonenumber

**MinFldArgs, MaxFldArgs**

1,1

**Number of standard roles**

1

**Min Length, Max Length**

8,8

1. All non-digits are filtered out of the attribute.
2. If the length of the attribute is less than MinLength(8), it is treated as ANON.
3. If the length is greater than MaxLength(8), the last eight digits are returned.
4. If there is an ANON table specified, the attribute is checked against the table. If the attribute exists, the value is treated as ANON.
5. If an EQUI table is specified, the attribute is checked. If the attribute does not exist in the table, then the value is treated as ANON.

**INTPHONE1**

This function standardizes international phone numbers by analyzing international calling codes.

**Output Type**

any

**Fldargs**

phicc,pharea,phonenumber

**MinFldArgs, MaxFldArgs**

3,3

**Number of standard roles**

1
1. The international calling code, area code, and the phone number are required fields to this function and the function first checks to make sure that the number of inputs are correct. If you do not configure the function correctly by leaving out one of these required fields, the Engine cannot start.

2. The non-digits are eliminated as a first filtering step.

3. The phone number field is then checked to see if there is an international dialing prefix (for example, +44) and if present, this is trimmed off. Then the country calling code is checked to verify that it is present in the phone number (by checking the valid list of country calling codes). If the code is present, it is also trimmed from the phone number field.

4. If there is an international dialing prefix in the area code field, it is trimmed off the string. The country code is also checked and, if present, is also trimmed.

5. If the above two steps do not yield a prefix or country code, the international calling code field is checked and the country code output from is used for processing.

6. After removing the prefixes and country codes, any long distance characters (0 or 1) are removed from the start of the remaining phone number.

7. The country calling code from the above steps is used to determine the country to which the phone number belongs.
   - 1 - US phone
   - 33 - French phone
   - 44 - UK phone
   - 49 - German phone
   - Other - default phone standardization

8. The number is then checked for min and max lengths. These min and max values are based on the country of origin. For each country of origin, max length is defined to be the longest legal phone number consisting of long-distance prefix + local number. This does not include the long-distance access numbers (for example, the 0 or 1 at the beginning of a phone number). The MinLength is the shortest legal local number and does not include any in-country access codes.
   a. If the phone string is less than minLength, the phone number is treated as an anonymous value.
   b. If the phone string is less than or equal to the maxLength, the remaining string phone string is returned.
   c. If the phone string is greater than maxLength, the left most maxLength characters are returned.

9. The number is also checked to see if it is present in the ANON table and is treated accordingly.
10. The standard role (stdrole) 2 is used for bucketing. The last six digits of the phone number are used.

INTPHONE1 example:

+44 (0) 282 999 7788 becomes 2829997788

PHONE1
This function is used to format a U.S. seven-digit phone number, not including area code.
Output Type
numeric

Fldargs
    phnumber

MinFldArgs, MaxFldArgs
    1,1

Number of standard roles
    1

MinLength, MaxLength
    7,7

1. If there is a leading 1, it is stripped off the phone number and any non-digits are ignored.
2. If the number of digits is greater than or equal to 10, then digits 4 through 10 are taken.
3. If the attribute is empty or less than MinLength (7) or greater than MaxLength (7), it is considered ANON.
4. If there is an ANON table specified in mpi_dvdxstd, the value is checked to see if it is present in the table. If found, the attribute is treated as ANON.
5. If there is an EQUI table specified in the mpi_dvdxstd, the attribute is checked to see if it belongs to this table. If it does not, the attribute is treated as ANON.
6. The attribute is then added to the comparison string.

PHONE1 example:

512-999-0456 is converted to 9990456
(1)512-999-0456 is converted to 9990456
512-999-045678 is converted to 9990456
(001)512-999-0456 is converted to 9990456

PHONE2

The PHONE2 function is similar to PHONE1, with the exception that PHONE2 allows more numbers in the fldargs.

Thus, if someone had an area code or extension number in the phone fields, this data could be processed using PHONE2. This function can also be used when the numbers have only the minimum of seven digits.

Output Type
    numeric

Fldargs
    phnumber, additional fields such as area code or extension are optional

MinFldArgs, MaxFldArgs
    1,3

Number of standard roles
    1

PHONEEND

This function standardizes phone numbers regardless of country format.
**Miscellaneous standardization functions**

There are miscellaneous standardization functions that handle abstract codes and strings that you want to pass through to the Master Data Engine.

**ABSCODE**

This function is used to handle any type of specialty (abstract) codes used by providers.

**Output Type**

any

**Fldargs**

attrval

**MinFldArgs, MaxFldArgs**

1,1

**Number of standard roles**

1

1. The attribute characters are first checked to see if they are alphanumeric and are converted to uppercase.
2. All special characters are eliminated.
3. If there is a missing attribute (length of 0), it is treated as an ANON value.
4. A lookup is performed for the attribute value in the mpi_strequi table by using SA<\text{n}>-ABS (where \text{n} is the source RecNo) and SA0-ABS values. If the value is found in the SA<\text{n}>-ABS or SA0-ABS tables, the equivalent code is replaced for the attribute. If matches are found in both tables, the <\text{n}> match is given preference. If it is not found, the attribute is treated as ANON.

5. The value returned as a result of step 3 is added to the compare string.

6. Since each source can have a different meaning for its specialty codes, it is essential to retain the source identity in the EQUI table.

ABSCODE example:

mpi_strequi table has the following entries:

SA0-ABS PSY 19
SA0-ABS NUR RNFU
....
SA7-ABS PSY 19

If you have an attribute value of 19, a lookup in the mpi_strequi table is performed and the specialty code of PSY is returned.

If you have an attribute value of Rnfu, all spaces are eliminated and the characters are converted to uppercase. Thus Rnfu becomes RNFU. A lookup for RNFU in the mpi_strequi table is performed and the code of NUR is returned.

**PASSTHRU**

This standardization function does not change the input before passing it through to the Master Data Engine processes.

This function was originally designed as a temporary workaround. Because it might not always perform as described, use of this function is discouraged. For example, if you attempt to standardize a value with certain special characters (like a colon, caret or period), you risk a negative impact on comparisons with the given member.

**Comparison functions**

These functions are used to compare data elements to each other and determine matches and scores.

Comparison functions work on the standardized data created during the data derivation process. Most of the functions use a weight table to make these determinations. All comparison functions use the mpi_cmphead, mpi_cmpspec, and mpi_cmpfunc tables.

After the data has been standardized and the comparison (cmpd) string has been formed, the actual comparisons are performed by using a set of comparison routines.

Attributes can be either Active or Inactive. To choose the attributes that are used in the comparison, every attribute (Active or Inactive) from one member is compared...
to every other attribute of the other member. The best value of the comparison is chosen. You can configure the algorithm to omit inactive attributes if you want.

The maximum number of comparison roles (cmp roles) allowed is 32.

**Token comparison**

It can be helpful to understand the logic used by the Master Data Engine to compare two tokens.

**Initials match**

If both tokens are initials, there is an exact match and the exact match weight for the initials is applied. If one token is an initial and the other is not and the first characters match, the weight is the exact match weight for the initial with the initial adjustment penalty.

**Full word comparison**

The exact match weight is applied if the words match completely.

**Phonetic name comparison**

The words are checked for phonetic matching and, if they match, the weight is computed by subtracting the phonetic adjustment penalty from the smaller of the two exact match weights. If this weight is smaller than the minimum phonetic weight, the minimum weight is used.

**Nickname comparison**

If there is a NICKNAME table specified in the mpi_cmpspec table, the nickname comparison is performed. Two names match from the nickname table if they have a common nickname or the nickname of one matches the original name of the other.

If there is a nickname match, the weight is computed by subtracting the nickname adjustment penalty from the smaller of the two exact match weights. If this weight is smaller than the minimum initial weight, the minimum is used.

**Nickname-meta comparison**

Tokens can also match through a nick-meta (phonetic) match. Two names match in the nickname table if they each have a nickname that matches phonetically, or the nickname of one phonetically matches the original name of the other.

If there is a nick-meta match, the weight is computed by subtracting the nick-meta adjustment penalty from the smaller of the two exact match weights. If this value is less than the minimum nick-meta weight, the minimum value is used.

**Edit distance comparison**

Edit distance measures the similarity between two tokens by calculating the number of character insertions, deletions, or transpositions it would take to make the tokens match.

If Editdistance * edit-distance factor <= maximum (length of the two words), then the edit distance weight is applied. In other words, EditDistance <= 1/edit-distance factor.

Setting __EDITDIST_FACTOR = 5 means there is an edit distance match if the edit distance is <= 1/5 or 20% of the longest string length.

Setting __EDITDIST_FACTOR = 0 means that every string pair matches by edit distance (this setting is not suggested).
Setting __EDITDIST_FACTOR to a large number means that few if any pairs match by edit distance.

The weight is obtained by applying the edit-distance adjustment penalty to the exact weights, as in the previous cases. In other words, you are trying to see how much of an edit distance is considered for a match.

See “Edit distance comparison functions” on page 310 for more detail.

**Prefix and Compound comparison**

Both AXP and CXNM functions allow for prefix and compound word comparison and matching. By using PREFIX or COMPOUND weight generation parameters, match cases that might normally be missed in edit-distance processing can be accounted for and produce more accurate results. For example, “Cleveland” and “Cleve” are not close enough for an edit distance comparison to result in a match. Using a prefix parameter would adjust for this distance in the comparison. As well, the compound parameter would account for “WALMART” in one string and “WAL MART” in another.

**Word mismatch**

If none of the previous matches pass, then the words are completely different. In this case, a zero weight is assigned.

**Acronym comparison**

Using the acronym comparison, if “ITS” is compared to “INTEGRATED TECHNOLOGY SOLUTIONS,” there is a match.

The acronym comparison works in two ways. First the words in “ITS” are checked to see if there is an acronym in INTEGRATED TECHNOLOGY SOLUTIONS and vice versa.

Acronym matches are scored as initial matches. In the described example there are three initial matches: I-INTEGRATED, T-TECHNOLOGY, and S-SOLUTIONS.

**Address comparison functions**

Address comparisons are a valuable way to identify member matches and are vital for organizations that use the householding functionality.

The comparison functions use the output of the address standardization functions.

**AXP**

AXP (address and phone) performs edit distance, phonetic, and frequency-based analysis on address data and an edit distance analysis on phone number data.

AXP uses the output of the ADDR2 standardization functions. It uses the cmpargs argument in mpi_cmpspec to specify the type of phonetic function to be applied. The possible phonetic functions (cmpargs) are: METAPHONE, IDENTAPHONE, ARABPHONE, and PREFIXMAP.

The AXP address and phone comparison function is based on information content and similarity. If the address consists of street information and postal codes, this information is used for comparison. When the postal code is not present, the City, State (or city, country) is used in the comparison.

The AXP function (and “CXNM (comparison function)” on page 318) compares ordered name tokens. To account for match cases with prefixes or compound
names, which might be missed in normal edit distance processing, use PREFIX or COMPOUND weight generation parameters. These parameters help achieve accurate name matching.

AXP further accounts for addresses that contain unit information, such as APT, FLOOR, or SUITE. By using the string type of SET, which is defined in mpi_strset, the address input is broken up by unit boundaries.

In some instances, you might want to give greater emphasis (greater weight) to certain tokens in the address string. For example, you might want to weigh ZIP Code higher than Streetline. The use of ADDR_ parameters enable weighting of address tokens which affect the average weight used in the comparison. By using these parameters, scoring is not penalized by missing tokens. Use of the ADDR_ parameters affects the average weight (AVERAGEWGT) in the comparison.

The AXP comparison function uses four weight tables. The mpi_wgtsval table contains the exact weights for the individual tokens (street address lines, numbers, postal codes, city, and state) and the parameters used in the AXP comparison. The mpi_wgt1dim table contains the weight for numeric exact match values in the address line. The mpi_wgt2dim is the final lookup table to get the final weight (for an address and phone index combination). The mpi_wghthead table contains the weight definitions for calculating match scores.

Number of roles
2

Number of dimensions (in each role)
1

cmpargs
strcode containing unit indicators; the format is UNITTYPES=<strcode>

dvdargs
ALLOWHASH (Note that the dvdargs setting is made in the standardization function, not in the actual AXP comparison function.)

Weight table
mpi_wgt1dim, mpi_wgt2dim, mpi_wghthead, mpi_wgtsval

Other tables
mpi_cmpspec, mpi_cmphead, mpi_cmpfunc

The following steps describe the comparison.

1. The street address is divided into subcomponents of streetline, unit boundary, region (city and state) and postal code. The address is broken by unit boundary if a strcode for unittypes is defined. If the ALLOWHASH dvdarg is passed from standardization, any # (number sign) in the address is not treated as punctuation. Otherwise, it is treated as punctuation and is stripped from the string.

2. The information content for each of these subcomponents is calculated in both strings. The information content is the weight associated with each token and is computed as follows:
   • If the token type is ZIP code, the weight is taken from the 1DIMZIP table defined in mpi_wgt1dim.
   • If the token type is any other numeric, the weight is taken from the 1DIM table in mpi_wgt1dim.
   • If the token type is non-numeric, the weight is taken from the mpi_wgtsval table.
Using the example of “12345 PARK RN TN”, the weight for 12345 is looked up at index 6 in the 1DIMZIP table in mpi_wgt1dim (length of numeric data). Weights for PARK, RD, and TN are looked up in mpi_wgtsval.

3. The weight adjustment is done for both strings as follows. If the total streetline weight is greater than Address Street MaxWgt, the streetline weight and each individual token within the streetlines are adjusted to scale down within the MaxLimit. This process is repeated for region and postal code weights as well.

   For example:
   
   6001 PLAZA ST
   6001 - 10
   PLAZA - 20
   ST - 10
   
   Total streetline weight = 40
   AvgMaxWgt for streetline = 25
   
   The adjustment is as follows:
   
   6001 - 10 *25/40 = 6.25
   PLAZA - 20 *25/40 = 12.5
   ST - 6.25

4. If the address contains a unit, the weight adjustment is as follows.

   a. Unit starting positions are determined based on settings in mpi_strset. Only the tokens in the first cmpval dimension (stline components are examined.

   b. If the primary address components are equal and the unit components are unequal, then the weight is adjusted by the value of the _ADDR_UNIT_ADJWGT parameter. Otherwise, the addresses are equal or one/both did not contain unit information, so no adjustment is made. For example, use the addresses of:

   5001 Plaza on the Lake Blvd Suite 111 and 5001 Plaza on the Lake Blvd Suite 222

   These addresses would allow for the adjustment since they are equal in primary components (everything left of Suite) and unequal in unit components (everything right of Suite).

5. The comparison is performed iteratively, like CXNM comparisons, in a grid manner. The resulting weights in this comparison are the similarity weights. The tokens from the two street addresses being compared are checked for all possible matches (including compound name matching) as listed in the beginning of this section. At each grid comparison, the best match position is also noted. The CELL-DIFF_<n> (difference between table cells) is the positional penalty. The penalty is applied if the previous best match position and the current match are more than one cell apart. The resulting weight from this comparison is an input to the TOTALWGT.

6. Post-grid processing is performed in the following manner:

   a. If there is a postal code in both strings and if the weights of both postal code strings are greater than the addrPostalMin weight, then the postal code similarity weight (as calculated in step 4) is added to the TOTALWGT.

   The AVERAGEWGT is calculated by scaling the weights from two strings based on the ADDR_POSTAL_NORM.

   If AddrPostalNorm = 0, the minimum of the two string weights is added to the AVERAGEWGT.

   If AddrPostalNorm = 1, the maximum is added.

   For anything in between, the appropriate scaled weights are added.
b. If conditions from step 5.a fail, a check is made to see if there is a region specified in both strings and if the region weights are more than AddrRegionMin. If they are, the region similarity (as calculated in step 4) is added to the TOTALWGT.

The AVERAGEWGT processing is done in a manner like step 5.a using the AddrRegionNorm factor.

c. If there is not a “good” postal code or region (that is, step 5.b fails), then the (AddrRegionMinwgt + AddrPostalMinwgt)/2 is added to the AVERAGEWGT.

d. The Streetline part of the address is then checked. If there are at least two street tokens in both strings AND both are more than AddrStreetMin wgt, the similarity from Streetline gets added to the TOTALWGT.

The AVERAGEWGT is calculated as in step 5.a using the AddrStreetNorm factor.

e. If there is not "good" street data (step 5.d fails), then the AddrStreetMinwgt is added to the AVERAGEWGT.

7. The Normalized index is then calculated for Address. The Normalized Similarity = NormMaxIdx(15) * TOTALWGT/AVERAGEWGT.

Normalized Index = 15 - Normalized Similarity

If this Normalized Index is greater than NormMaxIdx, then that Normalized Index = NormMaxIdx.

If the Normalized Index is less than NormMinIdx, the Normalized Index = 1

This index is the ADDRESSINDEX.

8. The Phone comparison is done by using edit-distance compare. The string tokens are compared against each other and the best match (or minimum edit-distance) is taken. This distance is the PHONEINDEX (in the mpi_wgt1dim table, the edit-distance of 0 is index 1, and so on).

9. For 2dim weights, the AddressIndex and Phone Index are the inputs to look up the mpi_wgt2dim table for the FINAL weight for AXP.

AXP example:

The strings to be compared are:

6001 Plaza St, 78750
6001 New Plaza, 78751

Assume the following weights:

Plaza, 6001, St, New, 78750 = 10
Cell_diff_2 = 5
AddrPostalMinwgt = 9
AddrStreetMinWgt = 10
AddrPostalNorm = 0
AddrStreetNorm = 0.5

<table>
<thead>
<tr>
<th></th>
<th>6001</th>
<th>Plaza</th>
<th>St</th>
<th>78750</th>
</tr>
</thead>
<tbody>
<tr>
<td>6001</td>
<td>10*</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>New</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Plaza</td>
<td>10</td>
<td>10+10-5=15**</td>
<td>15 (Stline)</td>
<td>15</td>
</tr>
<tr>
<td>78750</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>15+10=25</td>
</tr>
</tbody>
</table>
1. Streetline: 6001 Plaza St
   Postal code: 78750
   Streetline: 6001 New Plaza
   Postal code: 78750

2. Information content:
   6001 Plaza St - 30
   78750 - 10
   6001 New Plaza - 30
   78750 - 10

3. Assume that this information content is within the maximum limits.

4. As seen above the comparison is token by token. 6001 and 6001 are the first
   compared tokens and they match exactly. This comparison is the best match so
   far (10*). As the comparison continues along Row 1, the best weight from the
   top, left, and diagonal cell is carried forward. Row 2 continues the same way
   since there is no match. In Row 3, the Plaza and Plaza match exactly, so the
   weight is the existing carried forward weight + exact match weight = 10+10 =
   20. BUT, the last best match was in row 1. The current match is in Row3. Since
   there is a celldiff of 2, the CELL_DIFF penalty is applied. So weight = 20 -5
   =15, which is the new best weight.
   StreetLine Similarity = 15
   Postal Code Similarity = 10 (Match wgt for 78750 and 78750)

5. Since Postal code and Streetline satisfy the conditions, TotalWgt = 15 + 10 = 25.
   StreetLine AverageWgt = (30+30)/2 = 30 (since AddrStreetNorm =0.5)
   PostalCode Average Wgt = Min(10, 10) = 10 (since PostalStreetNorm =1)
   AverageWgt = 30 + 10 =40


7. On the phone comparison, assume that the comparison gave an exact match.
   Thus, the edit-distance was 0, so the index is 1.

8. The 2dim lookup index is (14,1).

**AXP parameters**

There are two types of parameters used by AXP. The first is a set of parameters
used during the comparison step of weight generation. The second set of
parameters uses the outputs from weight generation and are used during the
actual data comparison to determine a final score.

**Weight generation comparison parameters:**

As previously mentioned, with normal edit-distance processing, there are names
that might not be recognized. Using the following weight generation parameters
for prefix and compound address matching can significantly improve comparisons.

For example, if string A contains “CLEVELAND” and string B contains “CLEVE,”
the strings are not close enough for the edit distance partial match. However, the
prefix parameter adjusts for this scenario.

When comparing compound names, if string A contains “WAL MART” and string
B contains “WALMART,” the parameters do allow for credit as a match.

Another example is string A with an address of “123 MAIN St SUITE 456” and
string B “123 MAIN ST SUITE456.” Using the compound parameters would allow
credit for a match.
These weight generation parameters include the following descriptions.

**Note:** These parameters are defined in the mpi_wgtsval table by using a wgtcode of PARM. When you generate your weights in Initiate IBM Initiate Workbench, you can right-click and select Get Weights. The contents of mpi_wgtsval are copied into your project directory and from there you can open and edit the file.

- **PREFIX_FACTOR** – The threshold for prefix match is defined by a configurable prefix match factor which is an integer PREFIX_MATCH_FACTOR. For example: Ni = “MICRO” and Mj = “MICROSOFT”
  If Ni matches the beginning of Mj and PREFIX_MATCH_FACTOR* len(Ni) >= len(Mj), then the match is a prefix match.
  Setting PREFIX_MATCH_FACTOR to 2 means that Ni should be at least half the length of Mj. Setting PREFIX_MATCH_FACTOR to 3 indicates that Ni should be at least one-third the length of Mj.
  The default value for PREFIX_MATCH_FACTOR is **2**.

- **PREFIX_ADJWGT** – If a prefix match is identified, the prefix_adjwgt is used to adjust the prefix token (for example, “MICRO”) weight in the following manner:
  (Prefix compare token)Ni_wgt = MIN(Ni_wgt, Mj_wgt) - PREFIX_ADJWGT
  The default value for PREFIX_ADJWGT is **100** or **1.0**.

- **PREFIX_MINWGT** – Used as a lower boundary for any prefix weight adjusted token. The prefix adjusted weight never falls below this weight value. The default value for prefix_MINWGT is **50** or **.5**.

- **PREFIX_MAXWGT** – Used as an upper boundary for any prefix weight adjusted token. The prefix adjusted weight never goes above this weight value. The default value for prefix_MAXWGT is **300** or **3.0**.

- **COMPOUND_ADJWGT** – A compound match is detected when comparing tokens “MICRO” “SOFT” versus “MICROSOFT”. The compound_adjwgt is used to adjust the compound token (“MICRO”) weight in the following manner:
  (Compound compare token “MICRO”)Ni_wgt = MIN(Ni_wgt, Mj_wgt) - COMPOUND_ADJWGT
  The default value for compound_adjwgt is **50** or **.5**.

- **COMPOUND_MINWGT** – Used as a lower boundary for any compound weight adjusted token. The compound adjusted weight never falls below this weight value. The default value for compound_MINWGT is **50** or **.5**.

- **COMPOUND_MAXWGT** – Used as an upper boundary for any compound weight adjusted token. The compound adjusted weight is never above this weight value. The default value for compound_MAXWGT is **400** or **4.0**.

**Comparison parameters:**

The following conditions and penalties are used in comparison to determine the final score.

**Boundary conditions.** AXP uses these parameters to check boundary conditions for Streetlines, Postal code, and Region.
- __ADDR_POSTAL_MAXWGT
- __ADDR_POSTAL_MINWGT
- __ADDR_REGION_MAXWGT
- __ADDR_REGION_MINWGT
- __ADDR_STREET_MAXWGT
- __ADDR_STREET_MINWGT
• __ADDR_UNIT_ADJWGT

The sum of the words in the streetline is checked to make sure that it does not exceed the ADDR_STREETLINE_MAXWGT. If it does, it is scaled to the MAX value. Similarly, the ADDR_POSTAL_MAXWGT and ADDR_REGION_MAXWGT are used for Postal and Region subcomponents of the Address.

If the streetline token weights are less than the ADDR_STREET_MINWGT, then the streetline weight does not contribute to the Avgwgt. Instead the ADDR_STREET_MINWGT is used.

Similarly if the Postal code and Region weights are less than their respective MINWGT, the average of the two MINWGTs contributes to the total average weight.

Position penalties. AXP uses these parameters to check for position penalties.
• __CELLDIFF_ADJWGT_2
• __CELLDIFF_ADJWGT_3
• __CELLDIFF_ADJWGT_4
• __CELLDIFF_MAXIDX
• __CELLDIFF_MINIDX

CELLDIFF parameters are applied during the comparison of one token to another. When two tokens that match have a cell difference of 2, the total weight is reduced by CELLDIFF_ADJWGT_2. The weight is checked to make sure that it is within the range limits.

CELLDIFF_MAXIDX is used to check that the cell difference does not exceed this Max.

Edit-distance. The following parameters are used to check the phone edit-distance. If it is equal to MCCIDX_EQUAL, then it is an exact match, otherwise it is a partial match.
• __DIST_MINIDX
• __DIST_MAXIDX
• __DIST_MCCIDX_EQUAL
• __DIST_MCCIDX_PARTIAL

If the edit-distance from phone number comparison is equal to DIST_MCCIDX_EQUAL, they are considered an exact match. Similarly, if the distance is equal to DIST_MCCIDX_PARTIAL, they are considered partial match.

Index calculation. AXP uses these parameters to calculate the index for 2dim tables. It also checks for EQUAL or PARTIAL matches (like the ones previously described).
• __NORM_MINIDX
• __NORM_MAXIDX
• __NORM_MCCIDX_EQUAL
• __NORM_MCCIDX_PARTIAL

Like the Phone parameters above, the NORM_MCCIDX values are used for the address equal or partial match.
NORM_MAXIDX and NORM_MINIDX are used to bind the Normalized value from the Address matching.

**Partial string matches.** These parameters are used for penalties for the partial string matches.

- EDITDIST_ADJWGT
- EDITDIST_FACTOR
- EDITDIST_MINWGT
- EDITDIST_MAXWGT
- FULLNAME_MAXWGT
- INITIAL_ADJWGT
- INITIAL_MINWGT
- INITIAL_MAXWGT
- NICKMETA_ADJWGT
- NICKMETA_MINWGT
- NICKMETA_MAXWGT
- NICKNAME_ADJWGT
- NICKNAME_MINWGT
- NICKNAME_MAXWGT
- PHONETIC_ADJWGT
- PHONETIC_MINWGT
- PHONETIC_MAXWGT

EDITDIST_FACTOR is used to determine if the two strings can use an edit-distance comparison. EDITDIST_ADJWGT is the penalty applied for an edit-distance match. The final weight is then checked to be within the ranges of the MIN and MAX values.

**Weight tokens.** The following parameters are used to give greater weight to tokens. Each parameter is a number between 0 and 1. Using 0 results in the average being equal to the minimum weight. Using 1 results in the average being equal to the maximum weight. Using 0.5 results in AXP behavior found in earlier versions of the software (8.5 and earlier) where the average is a true average. 0.5 is the default value. Here the average refers to the average weight for the two strings being compared.

- ADDR_STREET_NORM (this parameter allows you to control the weight of the streetline)
- ADDR_REGION_NORM (this parameter allows you to control the weight of the address region code)
- ADDR_POSTAL-NORM (this parameter allows you to control the weight of the postal code)

The following weight tables are expected for AXP:

- mpi_wgthead - the following entries must be present for AXP in the mpi_wgthead table

<table>
<thead>
<tr>
<th>Address Type</th>
<th>Weight</th>
<th>Address Component</th>
<th>CMPID</th>
<th>AXP, Dim, CMPID-AXP</th>
<th>AXP, Dim, CMPID-AXP</th>
<th>Dim</th>
<th>AXP, Dim, CMPID-AXP</th>
<th>Dim</th>
<th>AXP, Dim, CMPID-AXP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>1DIM</td>
<td>CMPID-AXP-1DIM</td>
<td>1DIM</td>
<td>CMPID-AXP-1DIM</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>1DIM</td>
<td>CMPID-AXP-1DIMZIP</td>
<td>1DIM</td>
<td>CMPID-AXP-1DIMZIP</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>2DIM</td>
<td>CMPID-AXP-2DIM</td>
<td>2DIM</td>
<td>CMPID-AXP-2DIM</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>AXP</td>
<td>CMPID-AXP-PARM</td>
<td>SVAL</td>
<td>CMPID-AXP-PARM</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>AXP</td>
<td>CMPID-AXP-XACT</td>
<td>SVAL</td>
<td>CMPID-AXP-XACT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
• mpi_wgtsval - mpi_wgtsval table for AXP should contain the *PARM tables and the *XACT tables. A partial sampling of these tables is listed here:

```
1|1|A|CMPPATIENT-PATADDRESS-PARM|__PREFIX_FACTOR|2|
1|1|A|CMPPATIENT-PATADDRESS-PARM|__PREFIX_MAXWGT|944|
1|1|A|CMPPATIENT-PATADDRESS-PARM|__PREFIX_MINWGT|50|
1|1|A|CMPPATIENT-PATADDRESS-XACT|22ND|831|
1|1|A|CMPPATIENT-PATADDRESS-XACT|26TH|822|
1|1|A|CMPPATIENT-PATADDRESS-XACT|28TH|825|
1|1|A|CMPPATIENT-PATADDRESS-XACT|35TH|831|
1|1|A|CMPPATIENT-PATADDRESS-XACT|36TH|808|
```

• mpi_wgt1dim - mpi_wgt1dim table should contain the *1DIM and *1DIMZIP tables. For example:

```
1|1|A|CMPPATIENT-PATADDRESS-1DIM|0|0|
1|1|A|CMPPATIENT-PATADDRESS-1DIM|1|794|
1|1|A|CMPPROVIDER-AXP-1DIMZIP|1|494|
1|1|A|CMPPROVIDER-AXP-1DIMZIP|2|267|
```

• mpi_wgt2dim - the final weights are read from the mpi_wgt2dim table. This is a sample of those weights:

```
1|1|A|CMPID-AXP-2DIM|0|0|482|222|59|31|-71|-83|0|0|0|0|0|0|0|0|
1|1|A|CMPID-AXP-2DIM|1|482|524|339|292|331|358|359|395|0|0|0|0|0|0|0|0|
1|1|A|CMPID-AXP-2DIM|2|359|455|295|280|281|258|257|289|0|0|0|0|0|0|0|0|
```

**USZIP (comparison function)**

The USZIP comparison function checks for a match on United States ZIP code.

The comparison result can be either a complete match on all five digits, a partial match on the first three digits, or a complete mismatch. Sometimes the three matched uncommon digits can have more weight than five matched common digits.

**Tip:** Use ZIP along with the Address and Phone in a three-dimensional compare function (as mentioned earlier) when possible.

**Number of roles**

1

**Number of dimensions**

(in each role)

1

**Weight table**

mpi_wgtval, mpi_wgthead

**Other tables**

mpi_cmpspec, mpi_cmphead, mpi_cmpfunc

1. If the first three digits match exactly, then the fourth and fifth digits are checked for matching. If these digits also match exactly, the corresponding five-digit exact-match weight is taken from the mpi_wgtval table.

2. If the first three digits match exactly and the fourth and fifth digits do not match, the partial weight for the three digits is taken from the mpi_wgtval table.

3. If even the first three digits do not match, it is considered a mismatch and the disagreement weight is taken from the mpi_wgtval table.

The following tables are expected for USZIP:

• mpi_wgthead
Attribute comparison functions

ATTR2S

ATTR2S can be used to compare any combination of two string-valued attributes.

ATTR2S replaces the previously used EXH function, but can extend beyond just eye and hair color comparison.

Weight generation support is also included, unlike EXH. The weight tables used by ATTR2S are different from the weight tables used by EXH. Because of this difference, it is necessary for you to perform weight generation when switching from EXH to ATTR2S. For height/weight comparisons, use HXW rather than ATTR2S.

Number of roles
2

Number of dimensions
(in each role)
1

Weight table
mpi_wgtsval, mpi_wgthead

Other tables
mpi_cmpspec, mpi_cmphead, mpi_cmpfunc

1. Values from cmprole1 and cmprole2 are compared for the two members.
2. If values are missing for both members, this comparison does not participate in scoring.
3. If values are available for both members for both cmproles:
   a. The values for both cmproles are equal. The weight used is from the BOTH table. The key used is “L+R” where L is the value from cmprole1 and R is the value from cmprole2. If no weight was generated for this key, the default agreement weight is used.
   b. Only the values from cmprole1 are equal. The weight used is from the LEFT table. The key used is the value from cmprole1. If no weight was generated for this key, the default agreement weight is used.
   c. Only the values from cmprole2 are equal. The weight used is from the RIGHT table. The key used is the value from cmprole2. If no weight was generated for this key, the default agreement weight is used.
   d. None of the values are equal. The weight used is from the BOTH table. The key is for disagreement (d).
4. If values are available for both members from only one cmprole:
   a. The values in the cmprole are equal. The weight used is from either the LEFT (cmprole1) or RIGHT (cmprole2) table. The key is the value for this cmprole. If no weight was generated for this key, the default agreement weight is used.
b. The values in the cmprole are not equal. The weight used is from either the LEFT (cmprole1) or RIGHT (cmprole2) table. The key is for disagreement (d).

Weight tables for ATTR2S:

- mpi_wgthead

<table>
<thead>
<tr>
<th>1</th>
<th>A</th>
<th>CMPID-ATTR2S-BOTH</th>
<th>SVAL</th>
<th>CMPID-ATTR2S-BOTH</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>SVAL</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-RIGHT</td>
<td>SVAL</td>
<td>CMPID-ATTR2S-RIGHT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- mpi_wgtsval table

<table>
<thead>
<tr>
<th>1</th>
<th>A</th>
<th>CMPID-ATTR2S-BOTH</th>
<th>a</th>
<th>95</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-BOTH</td>
<td>BLK+BLK</td>
<td>86</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-BOTH</td>
<td>BLU+BLU</td>
<td>86</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-BOTH</td>
<td>BLU+BRD</td>
<td>89</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-BOTH</td>
<td>d</td>
<td>-62</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-BOTH</td>
<td>GRN+PNK</td>
<td>56</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-BOTH</td>
<td>GRY+BLK</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-BOTH</td>
<td>MUL+PNK</td>
<td>92</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>a</td>
<td>115</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>BLK</td>
<td>109</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>BLU</td>
<td>79</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>d</td>
<td>-79</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>GRN</td>
<td>48</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>GRY</td>
<td>61</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>HAZ</td>
<td>68</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>MUL</td>
<td>115</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>PNK</td>
<td>130</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>a</td>
<td>116</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>BLK</td>
<td>17</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>BLU</td>
<td>87</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>BRO</td>
<td>90</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>COM</td>
<td>99</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>d</td>
<td>-21</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>PNK</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>RED</td>
<td>57</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>CMPID-ATTR2S-LEFT</td>
<td>WHI</td>
<td>120</td>
</tr>
</tbody>
</table>

Biometric comparison functions

HXW
The HXW function is used to compare height and weight together.

Number of roles
1

Number of dimensions
1

Height cutoff
2

Weight cutoff
5

Weight table
mpi_wgtsval, mpi_wgthead, mpi_wgt2dim, wgtnval

1. The comparison string values are actual height and weight values as standardized by HEIGHT and WEIGHT standardization functions. The height is in inches with upper and lower boundaries. Weight is in pounds with upper and lower boundaries.
2. When heights are compared, the differences between the actual heights are computed and a match is assumed if the difference is less than a threshold (value = 2).

3. When weights are compared, the percentage difference between the weights is calculated and it is assumed a match if the percentage difference is less than a threshold (value = 5).

4. The height and weight counts that are used for calculating the score (weight) is done by collecting the counts for height and weight intervals. Counts for height and weight alone are also calculated separately.

5. Since the height and weight counts are calculated by using intervals, the actual height and weight are used to declare a match. For example, one member can have a height of 79 and another a height of 80. These heights match because the difference is less than 2, but they belong to different intervals of 78 and 80. Two case examples are:

   a. Case 1 - Both height and weight are available and they match.
      1) Members are in the same interval for both height and weight. The weight is taken from the BOTH weight table by using quantized member height and weight.
      2) Members are not in the same interval for either height or weight. Take weights from the BOTH table for each member and add half of each to the final weight.

   b. Case 2 - Either height or weight alone is available and they match.
      1) Members are in the same interval for this attribute. The weight is taken from either the LEFT or RIGHT table (height or weight) by using the quantized member attribute.
      2) Members are in different intervals for this attribute. Weights are taken from the LEFT/RIGHT tables for each member and half of each is added to the final weight.

   c. Case 3 - At least one of the attributes is available but does not match. Weight is taken from the 2DIM weight table indexed by the quantized differences in height and weight.

   Tip: The wgtval table has values such as HGT72+WGT170 or MISSING+WGT170, or HGT72+MISSING which correspond to exact match for height 72 inches, weight 170 lbs, or height missing, exact match weight 170, or exact match height 72 inches, weight missing.

Weight tables for HXW:

- mpi_wgthead

- mpi_wgt2dim

- mpi_wgtval
ATTRFB
The ATTRFB comparison function is used to compare strings of unequal length.

The longer string is compared only with the number of characters found in the shorter string. The comparison results in a match, partial match, or no match.

Number of roles
1

Number of dimensions
(in each role)
1-dim

Weight table
mpi_wgthead, mpi_wgt1dim

Other tables
None

1. This function uses a one-dimensional weight table. The index 0 (zero) is reserved for missing data and typically has a weight of 0. Index 1 is reserved for a MATCH, index 1 for PARTIAL and index 3 for NO MATCH.

2. The number of characters in each string is determined.

3. If both strings are equal in length and match exactly, then the comparison is a MATCH.

4. If the strings are unequal in length, then the short string is compared with the front and with the back of the longer string (front-to-front and back-to-back). If the compared characters (either front-to-front or back-to-back) match, then the result is a PARTIAL match.

For example, by using the following strings:

String 1 = INITIATEIBM

String 2 = INITIATE

String 3 = IBM
Date comparison functions

**CDATE**

The CDATE comparison function is used with the CDATE standardization and bucketing functions to facilitate the comparison of incomplete, partially valid, or approximate date ranges.

The CDATE comparison function is based on the overlap between the two date ranges being compared.

**Number of roles**
- 1

**Number of dimensions**
- (in each role)
  - 1

**Weight table**
- mpi_wgthead, mpi_wgt2dim

1. The function uses a two-dimensional (2-dim) comparison function.
2. The function substitutes 07 for a missing month and 14 for a missing day. The substitution is used to avoid incorrect matches with missed anonymous dates.
3. If the dates are complete and exact, the dimensions are (1,1). If the dates ranges do not overlap, the dimensions are (15,15)
4. If the date ranges overlap:
   - Dimension 1 is based on how far apart the two dates are (diffDays).
   - Dimension 2 is based on precision. The precision is defined by the lower of the two resolution values (implying a more precise date).
   - The two-dimensional weight table indexes run from 0 to 15, in which (0, 0) signifies missing dates, (1, 1) signifies equal date ranges and (15, 15) signifies no overlap of date ranges. To handle overlapping dates, the precision and diffDays values must be quantized to fall between the 2-14 dimension range.

This table provides a feel for the indexes computed for the two dimensions given (diffDays and precision) in days.

<table>
<thead>
<tr>
<th>Number of days</th>
<th>Index</th>
<th>Approximate number of years</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>78</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>278</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>529</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>846</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>1245</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>1747</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>2379</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>3174</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>4176</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>5437</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>
**DOBA**

The DOBA function compares date of birth.

**Number of roles**
- 1

**Number of dimensions**
(in each role)
- 1

**Weight table**
- mpi_wgtnval, mpi_wgthead, mpi_wgt1dim

1. If the attributes contain the complete date of birth (YYYYMMDD), the DOBA function uses the comparison process from the "DATE (comparison functions)" on page 306.

2. If only the birth years are present (YYYY) and if the difference between them is less than or equal to 1, then the minimum of the difference is taken and subtracted from the current year. This process determines the current age. Using this value, the mpi_wgtnval table is checked to get the corresponding match. This result is considered an exact match as the difference between the birth years is less than or equal to 1.

3. If the difference between the birth years is greater than 1, they are considered different and the weight is obtained by looking up the difference divided by two (/2) in mpi_wgt1dim.

Example 1:

Compare 1988 and 1982. The difference is 6 (>1), hence they are different. The value used for lookup is 6/2 = 3. This value is looked up in wgtnval and the corresponding weight is returned.

Example 2:

Compare 1988 and 1989. The difference is 1 and, therefore they are equal. The value used for lookup is <current year> - 1988 = 18. This value is looked up in wgtnval and the corresponding weight is returned.

**Weight tables**

The following weight tables are expected

- mpi_wgthead

<table>
<thead>
<tr>
<th>CMPID-DOB-DIST</th>
<th>DIM</th>
<th>CMPID-DOB-DIST</th>
<th>CMPID-DOB-DIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- mpi_wgt1dim

<table>
<thead>
<tr>
<th>CMPID-DOB-DIST</th>
<th>CMPID-DOB-DIFF</th>
<th>CMPID-DOB-XACT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>182</td>
<td></td>
</tr>
</tbody>
</table>
DATE (comparison functions)
The DATE function is used to compare two dates.

Number of roles
1

Number of dimensions
(in each role)
1

Weight table
mpi_wgtval, mpi_wgthead, mpi_wgt1dim

1. If the lengths of the two strings are eight characters, then the year is extracted from both strings and the years are compared. If the years are equal, then the month and day are extracted and compared. If they are all equal, then the weight is computed by adding the weight for index1 from mpi_wgt1dim to the weight for this particular year from mpi_wgtval table. If the strings are less than or greater than eight characters, they are treated as ANON.

2. If the dates match exactly, or the edit-distance is zero, then the weight is a combination of two tables—mpi_wgtval and mpi_wgt1dim table.

3. If they do not match exactly, the corresponding edit-distance value is used as a lookup in the mpi_wgt1dim table. Here the mpi_wgtval table is not used. If the dates of one string match with the month of the other string and vice versa, it is a transposed match and the distance is treated as one.

DATE example:
1978-24-03 and 1978-24-03 have an exact match. Therefore, the weight will be the weight for index 1 in mpi_wgt1dim + weight for 1978 in the mpi_wgtnval table.

1978-24-03 and 1978-03-24 will have a single transposition and an edit distance of 1. Thus, the weight will be the weight for index 2 in the mpi_wgt1dim table.

1978-24-03 and 1979-24-03 have an edit-distance of 1 and the weight is the weight that corresponds to index 2 in mpi_wgt1dim.

**Weight tables**

The following are examples of the weight tables expected.

- **mpi_wgthead**
  
  | 1 | 1 | A | CMPID-DOB-DIST | 1DIM | CMPID-DOB-DIST | 5 | 0 | 0 | 0 |
  | 1 | 1 | A | CMPID-DOB-YEAR | NVAL | CMPID-DOB-YEAR | 0 | 0 | 0 | 0 |

- **mpi_wgtnval**
  
  | 1 | 1 | A | CMPID-DOB-YEAR | -1 | 394 |

- **mpi_wgt1dim**
  
  | 1 | 1 | A | CMPID-DOB-DIST | 0 | 0 |
  | 1 | 1 | A | CMPID-DOB-DIST | 1 | 0 |
  | 1 | 1 | A | CMPID-DOB-DIST | 2 | 182 |
  | 1 | 1 | A | CMPID-DOB-DIST | 3 | 18 |
  | 1 | 1 | A | CMPID-DOB-DIST | 4 | -190 |

**DATE2 (comparison function)**

DATE2 is intended for single date comparison, such as birthdays or any event that occurs on a specific day (for example, marriage or death).

DATE2 handles Date-Date, Date-Age, and Age-Age comparisons. If both input strings are eight digits, a Date-Date comparison is performed. If either string is a year only, an Age (Age-Age) difference comparison is performed. If one string is a date and the other string is a year age, a Date-Age comparison is performed. This function provides stronger standardization for data from sources that do not perform validation.

DATE2 uses six weight tables (four for date and two for ages).

**Number of roles**

1

**Number of dimensions**

(in each role)

1

**Optional cmpargs**

1) granularity, in percentage (default 5), and
2) maximum index in the age table (default 8)

**Weight table**

mpi_wgtnval, mpi_wgthead, mpi_wgt1dim

**Date-Date comparison:**

Date-Date comparisons use four weight tables:

1. **Year** – This table has exact match weights for the most common years, a default exact match weight for other years, and a year disagreement weight.
2. **MonthDay** – This table is used when both month and day match. It has four-digit month-day entries (for example, 0101, 0324, 0229).
3. **Month** – This table is used when months—but not days—agree. It has an exact match weight for all months and a month disagreement weight.
4. **Day** – This table is used when days—but not months—agree. It has an exact match for all days of the month (1-31) and a day disagreement weight.

The Year, Month, and Day are compared separately and the final date weight is computed by adding the year, month, and day weight.

For **Year** weights:
1. If either year is invalid, the year weight is 0.
2. If the years are equal, an agreement weight is used for that year in the weight table.
3. Otherwise, the year disagreement weight is used.

For **Month and Day**:
1. If both the month and the day match, the weight is derived from the MonthDay weights. Otherwise, the month and day weight is derived from the Month and Day weight tables.
2. If both the month and day do not match and a valid date can be formed by inverting the month and date of the first date, the date is inverted and compared to the second date.
3. The inversion process is repeated for the second date.
4. If either of inverted total weights is greater than the original month and day weight, this score, less a small penalty, is used. By default the penalty is 50. To adjust the default penalty, within the mpi_wgtnval table add a numval of -3 and the desired penalty.

**Age-Age comparison:**

The weight tables used are ACIRCA and AGE which are populated in the mpi_wgt1dim table.

The age difference weight table uses the percentage of age difference. For example, if the ages are 20 and 25, the age difference is computed as a percentage of the smallest age.

Two optional cmpargs can be used; 1) granularity, in percentage (default is 5), and 2) maximum index in the age table (default is 8). Granularity is configurable. For example if the granularity is 5, then the weight table has an entry for age differences between 0 and 5 percentages.

As mentioned previously, there are two age difference weight tables—circa age and precise age.

The circa age table is used when one or both of the ages is a circa age. When computed, a function is applied to the matched set age distribution to replicate the difference between real and guessed ages.

The precise age table is used when both ages are non-circa years.
The age comparison weight is computed by taking the difference in the birth years, computing the percent difference, and then looking up the difference in the correct weight table.

Weight tables for DATE2:

* mpi_wgthead

1 | CMPID-DOB-ACIRCA | 1 | CMPID-DOB-ACIRCA | 9 | 0 | 0 | 0 |
1 | CMPID-DOB-AGE   | 1 | CMPID-DOB-AGE   | 9 | 0 | 0 | 0 |
1 | CMPID-DOB-DAY   | NVAL | CMPID-DOB-DAY   | 0 | 0 | 0 | 0 |
1 | CMPID-DOB-MONTH | NVAL | CMPID-DOB-MONTH | 0 | 0 | 0 | 0 |
1 | CMPID-DOB-MONTHDAY | NVAL | CMPID-DOB-MONTHDAY | 0 | 0 | 0 | 0 |
1 | CMPID-DOB-YEAR   | NVAL | CMPID-DOB-YEAR   | 0 | 0 | 0 | 0 |

* mpi_wgt1dim – There are two tables here for DATE2.

1 | CMPID-DOB-ACIRCA | 0 | 0 |
1 | CMPID-DOB-ACIRCA | 1 | 100 |
1 | CMPID-DOB-ACIRCA | 2 | 50 |
1 | CMPID-DOB-ACIRCA | 3 | 25 |
1 | CMPID-DOB-ACIRCA | 4 | 0 |

......

1 | CMPID-DOB-AGE | 0 | 0 |
1 | CMPID-DOB-AGE | 1 | 550 |
1 | CMPID-DOB-AGE | 2 | 288 |
1 | CMPID-DOB-AGE | 3 | 163 |
1 | CMPID-DOB-AGE | 4 | 51 |

* mpi_wgtval – There are three tables for DATE2 here.

1 | CMPID-DOB-DAY | -2 | -70 |
1 | CMPID-DOB-DAY | -1 | 0 |
1 | CMPID-DOB-DAY | 1 | 136 |
1 | CMPID-DOB-DAY | 2 | 138 |

......

1 | CMPID-DOB-MONTH | -2 | -84 |
1 | CMPID-DOB-MONTH | -1 | 0 |
1 | CMPID-DOB-MONTH | 1 | 102 |
1 | CMPID-DOB-MONTH | 2 | 105 |
1 | CMPID-DOB-MONTH | 3 | 101 |
1 | CMPID-DOB-MONTH | 4 | 103 |
1 | CMPID-DOB-MONTH | 5 | 101 |

......

1 | CMPID-DOB-MONTHDAY | -2 | -57 |
1 | CMPID-DOB-MONTHDAY | -1 | 0 |
1 | CMPID-DOB-MONTHDAY | 1 | 242 |
1 | CMPID-DOB-MONTHDAY | 2 | 259 |
1 | CMPID-DOB-MONTHDAY | 3 | 259 |
1 | CMPID-DOB-MONTHDAY | 4 | 258 |
1 | CMPID-DOB-MONTHDAY | 5 | 260 |

**VR1D1C_DATE**

VR1D1C_DATE is a values-based, one-dimensional (1Dim) function.

VR1D1C_DATE uses one cmprole that returns a true edit-distance result. This function is a replica or alias of the “DATE (comparison functions)” on page 306.

**VR1D1C_DOBA**

VR1D1C_DOBA is a values-based, one-dimensional (1Dim) function.

VR1D1C_DOBA uses one cmprole that returns a true edit-distance result. This function is a replica or alias of the “DOBA” on page 305.
**Edit distance comparison functions**

Edit distance functions compare two strings and determine the number of insertions, deletions, or transpositions it would take to make the two strings the same.

For example, you have a three-dimensional (3Dim) comparison for \( x \) (ZIP + address + phone) to \( y \) (ZIP + address + phone). The edit distance is the number of edits it would take to obtain an exact match of the \( x \) and \( y \) strings.

You can have one-dimensional, two-dimensional, three-dimensional, or four-dimensional comparisons. Each dimension corresponds to a wgtdim table (mpi_wgt1dim, mpi_wgt2dim, mpi_wgt3dim, and mpi_wgt4dim).

There is a common set of numbers used in edit distance results.

- A 0 (zero) edit distance means that one string is missing from the comparison input.
- 1 means an edit distance of 0 and both strings match exactly.
- 2 means that there is an edit distance of 1, as in one edit (insertion, deletion, or transposition) must be made to make the strings match.
- 3 means that two edits must be made for a match, and so on.
- >10 means that you have a mismatch.

Most of the time, you can deduce the edit distance functionality by understanding how they are named. Names are formatted as DR<\( r \)>D<\( d \>><\( t \)> where:

- DR<\( r \)> means edit Distance Role and the <\( r \)> indicates the number of roles (or standardized attributes) that are being compared. This is typically a number between 1 and 4. If you are comparing phone number, you would have one role. If you are comparing ZIP + address + phone, you would have three roles.
- D<\( d \)> represents Dimensions with <\( d \)> being the number of dimensions (or tokens/fields that make up an attribute) that are being compared in each role.
- <\( t \)> is a letter representing the type of comparison being performed. Options are:
  - A = simple edit distance comparison resulting in match or no match, so each dimension of the weight table has exactly two entries.
  - B = quick edit distance returns an integer representing the edit distance. The returned edit distance can sometimes be higher than the true edit distance, but it is much more efficient. This comparison is suggested for long strings.
  - C = a real or true edit distance. This is the most accurate comparison, but can use system resources. This comparison is suggested for short strings or in instances where absolute accuracy is vital.

Using the example of DR3D1A, you can determine that this comparison is a distance-based comparison that is using three cmproles, one dimension per role, and results in a simple match-no match.

**Tip:** If you have performance concerns, you might want to avoid using an edit distance calculation for too many attributes in your algorithm.
DR1D1A, [DR1D1B, DR1D1C]

DR1D1A, DR1D1B, and DR1D1C functions are used to compare string data.

Attention:  DR1D1B/C, DR2D1B/C, DR3D1B/C, DR4D1B/C accept 1, 2, 3, and 4 parameters which specify the maximum edit distance for each role. For example, if the parameters are 4, 5, 6 for DR3D1C, and if the edit-distance of role 1 is greater than 4, it is truncated to 4.

The result of a DR1D1A comparison is either MATCH or NON_MATCH.

The result of a DR1D1B and DR1D1C is an integer that represents how close the two strings are to each other. This integer is called edit-distance and is used for distance-based comparisons. Examples include SSN, and ID Number. DR1D1B uses a quick edit-distance comparison and DR1D1C uses a full edit-distance comparison. DR1D1B is faster; however it is not as accurate as DR1D1C.

Number of roles
1

Number of dimensions
(in each role)
1

cmpargs
edit-distance limit

Weight tables
mpi_wgthead, mpi_wgt1dim

The weight table for all three comparisons is a one-dimensional table (1dim). The index 0 (zero) is reserved for missing data and typically has weight of 0. Index 1 is an exact match. For DR1D1A, index 2 is a non-match. For DR1D1B and DR1D1C the similarity of the strings decreases as the index increases.

1. The two strings are compared by using an edit-distance function iteratively, first looping through one string and then the other. This step produces the edit distance required to transform one string to match the other. Transformations are deletion, insertion, addition, or transposition. For example:
   • PETER and PETED have an edit distance of 1 as it takes one-character change.
   • PETER and PETE have an edit distance of 1, as there is one insert to change one to another.
   • PETER and PETRE have an edit distance of 1, as it takes a transposition of R and E to change one to another.
   • PETER and PEARE have an edit distance of 2, as A must change to T and R and E undergo transposition.

Note:  When the function returns a distance of 1 that implies an edit distance of 0 or an exact match. When it returns 2, it is an edit distance of 1, and so on. If the function returns a 0, then one of the strings is missing, <

2. If there is a limit specified in the cmpargs field then that limit is used. If not, the edit-distance from the previous step is used for the distance.

3. If the distance is 0, the weight is 0 (one or both of the strings are missing).

4. If the distance is 1, they are considered an exact match and the weight is taken from the mpi_wgt1dim table (see the wgt1dim table example).
5. If the distance if anything else, they are considered different and weight is taken corresponding to this distance from the weight table (see the wgt1dim table example).

wgt1dim table example:

CMPID-SSN-DIST 0 0 => means missing values

CMPID-SSN-DIST 1 605 => edit distance of 0

CMPID-SSN-DIST 2 415 => edit distance of 1

....... 

**DR2D1A, [DR2D1B, DR2D1C]**

These comparison functions work the same way as the DR1D1A/B/C functions, except that they use a two-dimensional weight (wgt2dim) table. These functions are often used for combinations like "last name + ZIP" where the strings are fairly short.

**Attention:** DR1D1B/C, DR2D1B/C, DR3D1B/C, DR4D1B/C accept 1, 2, 3, and 4 parameters which specify the maximum edit distance for each role. For example, if the parameters are 4, 5, 6 for DR3D1C, and if role1 has an edit-distance greater than 4, it is truncated to 4.

**Number of roles**

2

**Number of dimensions**

(in each role)

2

**cmpargs**

limit1, limit2

**Weight tables**

mpi_wgthead, mpi_wgt2dim

As in the previous case, the difference between the A, B, and C functions is the way the edit-distance works. DR2D1C is the better of the edit-distance functions, which also takes more time to run than the other two.


To compare this Address and ZIP code in the same function, use the DR2D1A function.

The edit-distance of Address is 2, and the edit distance of ZIP code is 0 (exact match).

A lookup in mpi_wgt2dim is performed with the indexes (3,1), because the table lookup always adds 1 to the edit-distance values.

**Reading a 2dim table**

This table shows a sample wgt2dim table of Address and Phone.

<table>
<thead>
<tr>
<th>WgtCode</th>
<th>wgtidxno</th>
<th>wgtval0</th>
<th>wgtval1</th>
<th>wgtval2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPID-AXP-DIST</td>
<td>0</td>
<td>0</td>
<td>332</td>
<td>331</td>
</tr>
</tbody>
</table>
To look up the value of (3,1), first locate 3 in the wgtidxno column. Now, move along this row to get the second index 1 (wgtval1). The weight value is 190.

**DR3D1A, [DR3D1B, DR3D1C]**

These functions use a three-role comparison and three-dimensional weight (wgt3dim) table.

**Attention:** DR1D1B/C, DR2D1B/C, DR3D1B/C, DR4D1B/C accept 1, 2, 3, and 4 parameters which specify the maximum edit distance for each role. For example, if the parameters are 4, 5, 6 for DR3D1C and if the edit distance for role 1 is greater than 4, it is truncated to 4.

**Number of roles**
3

**Number of dimensions**
(in each role)
1

**Weight table**

| mpi_wgthead, mpi_wgt3dim |

If you use ZIP code, Address and Phone for comparison, you can use the three-dimensional edit-distance functions and refer to the wgt3dim weight table.

If the ZIP code returns a distance of 3, Address returns 2 and Phone returns 0, the value for (3,2,0) is looked up in the wgt3dim table.

This means that the ZIP code has an edit-distance of 2, Address has an edit-distance of 1, and Phone is missing. Remember, the lookup always adds 1 to the edit distance values.

**Reading a 3dim table**

<table>
<thead>
<tr>
<th>Wgtcode</th>
<th>wgtidx1</th>
<th>wgtidx2</th>
<th>wgtval0</th>
<th>wgtval1</th>
<th>wgtval2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPC-ZAP-DIST</td>
<td>0</td>
<td>0</td>
<td>334</td>
<td>332...</td>
<td></td>
</tr>
<tr>
<td>CMPC-ZAP-DIST</td>
<td>0</td>
<td>1</td>
<td>232</td>
<td>221</td>
<td>220...</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMPC-ZAP-DIST</td>
<td>3(*)</td>
<td>0</td>
<td>101</td>
<td>100</td>
<td>99...</td>
</tr>
<tr>
<td>CMPC-ZAP-DIST</td>
<td>3(*)</td>
<td>1(#)</td>
<td>100</td>
<td>99</td>
<td>97...</td>
</tr>
<tr>
<td>CMPC-ZAP-DIST</td>
<td>3(*)</td>
<td>2</td>
<td>99</td>
<td>98</td>
<td>.....</td>
</tr>
</tbody>
</table>

An edit distance of (2,0,1) for ZIP code, Address, and Phone refers to an index of (3,1,2).

In the 3-dim table, find the first index 3 in the wgtidx1 column(*). Within this index 3, find the value 1 in the second column wgtidx2(#). To get the last index 2, move along this same row until the column with wgtval2 is reached. The value in this column is the weight (in this case, 97).
**DR4D1A, [DR4D1B, DR4D1C]**

These functions use a four-role compare and four-dimensional weight (wgt4dim) table. They behave in the same way as the three-dimensional compare functions.

**Attention:** DR1D1B/C, DR2D1B/C, DR3D1B/C, DR4D1B/C accept 1, 2, 3, and 4 parameters which specify the maximum edit distance for each role. For example, if the parameters are 4, 5, 6 for DR3D1C, and if the edit-distance of role 1 is greater than 4, it is truncated to 4.

**Number of roles**
4

**Number of dimensions**
(in each role)
1

**Weight table**
mpi_wgthead, mpi_wgt4dim

**Reading a 4dim table**

<table>
<thead>
<tr>
<th>Role</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

The wgt4dim table is read in the same way as a wgt3dim table, however it has an extra column.

To get an index of (0,0,2,3), look for 0 in column 1, then move along the same row and look for 0 in column 2. Move along this row and look for 2 in column 3, and then look for column number 3 along this row.

**DR1D1A_SRC, [DR1D1B_SRC,DR1D1C_SRC]**

These are distance-based comparison functions that are used in a similar manner as DR1D1A. The difference is that the source code is added to the string before it is compared.

Many systems use the same numbering conventions when assigning account or record numbers. This convention can increase the likelihood of two different persons receiving identical numbers from different systems. For example, account number 4231 from SRC_A is not necessarily the same person as account number 4231 from SRC_B. By appending the source code to the number (SRCA4231 and SRCB4231) creates “distance” between the string contents.

The functions work in the same way as DR1D1* functions, except that they use the source values from the two comparison strings to check for that source weight value in mpi_wgtsval table. The corresponding attributes are compared only when this weight value is not 0.

The DR1DRA_SRCB and C differ in the way the edit-distance is computed, as in the earlier cases.

**Tip:** You might also see functions with “SRC” at the beginning, such as SRCDR1D1A. These functions (SRCDR1D1A, B, C) behave exactly as DR1D1A, B, C.
Equivalent comparison functions

Equivalent comparisons look for exact matches and are often used for comparing attributes like gender, eye color, and birth year.

There are two equivalent comparison functions: EQVD and EQVN.

**EQVD**

EQVD is a simple string comparison, used for a simple MATCH or NO-MATCH type of comparison with different match weights for different match values.

**Number of roles**

1

**Number of dimensions**

(in each role)

1

**Weight table**

mpi_wgtsval, mpi_wgthead

**Other tables**

mpi_cmpspec, mpi_cmphead, mpi_cmpfunc

1. A comparison is performed between the two strings.
2. If the strings match exactly, then the weight value for that particular string is looked up in mpi_wgtsval. If the value is not found, the default agreement weight is taken.
3. If the strings do not match, then the weight value corresponding to the disagreement is looked up in mpi_wgtsval.

**EQVD example:**

If you are using EQVD to compare gender, then the result can be an exact match with M, exact match with F or no match. The example shows a mpi_wgtsval table:

<table>
<thead>
<tr>
<th>CMPB-SEX-XACT</th>
<th>M</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPB-SEX-XACT</td>
<td>F</td>
<td>50</td>
</tr>
<tr>
<td>CMPB-SEX-XACT</td>
<td>a</td>
<td>0</td>
</tr>
<tr>
<td>CMPB-SEX-XACT</td>
<td>d</td>
<td>0</td>
</tr>
</tbody>
</table>

If you want to compare:

- M with F - look up row4 for disagreement
- M with J SMITH, M - look up row1 for M weight

**EQVN**

EQVN uses a numeric value comparison.

EQVN behaves in the same way as EQVD, with the exception of the numeric value comparison. EQVN uses mpi_wgtnval table for the lookup.

False Positive comparison functions

False positive filters are used to filter information, rather than function as normal comparison functions.

False positive filters are used to ascertain whether the records do indeed represent the same person rather than members belonging to the same family.
**FPF**

The FPF (false positive filter) comparison function is a filter that assists in determining whether the records being compared represent the same person rather than members belonging to the same family.

The FPF comparison function is used as prerequisite of XNM (Name), DOB (Date) and SEX (Gender) as the comparison speccodes. FPF does not use a comprole. The cmpseqno for FPF should be higher than the setting of XNM, DOB, and SEX.

The rules for FPF are as follows:
1. If there is a complete mismatch on the name, the FPF is set to true.
2. If there is a phonetic or partial name match AND either DOB or SEX disagrees, the FPF is set to true. FPF looks at DOB or SEX only if there is a partial match.
3. If a name suffix disagrees, the FPF is set to true.
4. ANON values are treated as a 0 in comparisons and are treated neutrally by FPF.

The weight for FPF is looked up in the mpi_wgt1dim table. The entry for index 0 has a value of 0, which implies that FPF is not true. An Index value of 1 implies FPF is set to true and has a weight value (normally around -10.0).

**FPF2**

FPF 2 is a second False Positive Filter that uses a four-dimensional weight table (mpi_wgt4dim) that can be configured. Generally, FPF2 is a better option than FPF.

The prerequisites are all or any subset of the following: birth year differences, birth date edit distance, gender, SSN edit distance, and name. The cmpspecodes that should be present are: DOB, SSN, SEX, and XNM.

**Reading FPF tables**

In this table, dimension 1 is Name and Gender result.

<table>
<thead>
<tr>
<th>Dim 1 Index</th>
<th>Name Result</th>
<th>Gender Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Missing</td>
<td>Missing</td>
</tr>
<tr>
<td>1</td>
<td>Exact</td>
<td>Missing</td>
</tr>
<tr>
<td>2</td>
<td>Partial</td>
<td>Missing</td>
</tr>
<tr>
<td>3</td>
<td>Disagree</td>
<td>Missing</td>
</tr>
<tr>
<td>4</td>
<td>Missing</td>
<td>Agree</td>
</tr>
<tr>
<td>5</td>
<td>Exact</td>
<td>Agree</td>
</tr>
<tr>
<td>6</td>
<td>Partial</td>
<td>Agree</td>
</tr>
<tr>
<td>7</td>
<td>Disagree</td>
<td>Agree</td>
</tr>
<tr>
<td>8</td>
<td>Missing</td>
<td>Disagree</td>
</tr>
<tr>
<td>9</td>
<td>Exact</td>
<td>Disagree</td>
</tr>
<tr>
<td>10</td>
<td>Partial</td>
<td>Disagree</td>
</tr>
<tr>
<td>11</td>
<td>Disagree</td>
<td>Disagree</td>
</tr>
</tbody>
</table>

- Dim 2 is DOB Edit Distance:
  - Index 0 is DOB missing
  - Index 1 is edit distance 0
• Index 2 is edit distance 1, and so on
• Dim 3 is Birth Year Difference:
  – Index 0 – DOB missing
  – Index 1 - diff is 0-4
  – Index 2 - diff is 5-9
  – Index 3 - diff is 10-14
  – Index 4 - diff is 15-19
  – Index 5 - diff is 20 or greater
• Dim 4 is SSN Edit Distance:
  – Index 0 – SSN missing
  – Index 1 – edit distance 0
  – Index 2 – edit distance 1, and so on

The mpi_wgt4dim table is used as the lookup for the FPF2. Here the index from the above rules results is looked up and the corresponding weight is used. For example:
  • Entry 6, 2, 5, 4 is:
    – Partial Agreement on Name
    – Agreement on Gender
    – Edit Distance 1 on DOB
    – Difference 20 or greater on DOB
    – Edit Distance 3 on SSN

Geographic comparison functions

GEO (comparison function)
The GEO comparison function compares locations by calculating the distance between them.

Locations closer together are considered more similar than locations farther apart.

Number of roles
  1

Number of dimensions (in each role)
  1

dvdargs
  Resolution - specified as meters

Optional cmpargs
  resolution to be used in meters

Weight tables
  mpi_wgt1dim, mpi_wgthead

Input for the GEO comparison function must come from an attribute standardized by the GEO standardization function. See the standardization function for "GEO (standardization function)" on page 271 for the details of the input and standard formats.
The GEO comparison operates by calculating the great-circle distance between the two locations being compared. The distance is calculated in meters. This distance is converted into a similarity measure by taking the base-10 logarithm according to the formula:

\[
\text{Similarity} = 2 \times \log_{10}(\text{distance} / \text{resolution})
\]

For distances less than the resolution, the similarity is set to 0. The default resolution is 1 meter. The resolution can be changed by providing an argument to the comparison function. The argument specifies the resolution to be used (in meters).

The GEO comparison function uses a single 1Dim weight table. The weights are indexed by the similarity measure described. The similarity is offset by 1 so that index 0 can be used for missing data.

**Name comparison functions**

Name (XNM) comparisons use various comparison techniques, such as phonetic, nicknames, substitutions, and edit distance.

Like the AXP function, the cmpargs in mpi_cmpspec are used to specify the type of phonetic function to use. Options are METAPHONE, IDENTAPHONE, ARABPHONE, or PREFIXMAP.

Also like AXP, the XNM functions use parameters for weight generation, position penalties, edit distance, index calculation, and other parameters. These parameters are described in the specific XNM functions.

**BXNM (comparison function)**

The BXNM comparison is used to compare provider names.

This function uses one cmprole, 1Dim, and cmpargs to specify the type of metaphone algorithm to be used. It works in the same way as "PXNM (comparison function)" on page 323.

**CXNM (comparison function)**

The CXNM comparison routine is used for business name comparisons.

CXNM compares ordered name tokens. As a reminder, cmpargs are used to specify the type of phonetic function to be applied.

Greater accuracy can be achieved by using PREFIX or COMPOUND weight generation parameters. Without use of these parameters, match cases with prefixes or compound names might be missed in normal edit distance processing.

**Tip:** You can use CXNM without phonetics for ideographic languages, such as Japanese.

**Number of roles**

1

**Weight table**

mpi_wgtsval, mpi_wgthead
Weight generation parameters

PREFIX_FACTOR, PREFIX_ADJWGT, PREFIX_MINWGT,
PREFIX_MAXWGT, COMPOUND_ADJWGT, COMPOUND_MINWGT,
COMPOUND_MAXWGT

The comparison function compares the two strings in an iterative manner, first looping through one of the compare strings and then through the other. The tokens are compared against one another during each iteration.

In business name parsing, the order of the words is important. Every time there is an order displacement, there is a penalty associated with the next match. For example: INITIATE SYSTEMS LTD and INITIATE SYSTEMS LTD match better than INITIATE SYSTEMS LTD and INITIATE LTD SYSTEMS.

The result of each token match could be one of the following:

- Initial match
- Partial match
- Phonetic match
- Acronym match
- Nickname match
- Nickname_meta match
- Prefix/compound match
- Edit distance match
- Total name mismatch

The comparison logic for two strings, both with a set of words/tokens, is as follows:

1. Token-by-token comparison takes order into consideration. As such, if the strings “CLEVELAND CLINIC” and “CLINIC OF CLEVELAND” are compared, only one of the two tokens CLEVELAND and CLINIC would match.

2. Penalties are applied for extra tokens between matches. If the strings “JIMS TRUCKS” and “JIMS PRETTY BIG TRUCKS” are compared, the tokens JIMS and TRUCKS would match. However, there would be a penalty applied to the second match, because of the extra tokens between JIMS and TRUCKS.

3. The prefix and compound name matching is done for CXNM. A check is done to see if the two tokens are compound matches (for example, INITIATE SYSTEMS and INITIATESYSTEMS).

a. First the strings are checked to see if they are numeric, since no prefix or compound match is done for numerics.

b. If the strings are a compound match, the weight of the non-compound string is taken and the compound adjustment weight is subtracted. This weight is then checked to see that it is within the min and the max ranges.

c. Prefix matching is then checked. If string 1 matches at least the string 2 length/prefix factor, then string 1 is considered a prefix of string 2. PREFIX_FACTOR is currently set to a default value of 2.

d. The PREFIX_MATCH weight is then applied by subtracting the PREFIX_ADJWGT (prefix adjustment) weight from the min (string1 and string2) weights. This weight is then checked to be within the min and max range. If it is not within the range, the weight is limited to the min/max as appropriate.

4. After all the comparisons have been done, the scores for all the matched tokens are added up, and then divided by the average of the two total token weights.
This percentage is then normalized to a score from 0 to the maximum normalized index, which is typically 16. This index is used to look up the final weight in the PARM weight table.

5. This normalized weight is compared to the different thresholds and the comparison output is MATCH, PARTIAL MATCH or MISMATCH.

**CXNM parameters**

There are two types of parameters used by CXNM. The first is a set of parameters used during the comparison step of weight generation. The second set of parameters use the outputs from weight generation and are used during the actual data comparison to determine a final score.

**Weight generation comparison parameters**

As mentioned previously, with normal edit distance processing, there are names that might not be recognized. Using the following weight generation parameters for prefix and compound address matching can significantly improve comparisons.

For example, if string 1 contains “CLEVELAND” and string 2 contains “CLEVE,” the strings are not close enough for the edit distance partial match. However, the prefix parameter adjusts for this scenario.

In the case of compound names, if string 1 contains “WALMart” and string 2 contains “WALMART,” the parameters do allow for credit as a match.

Weight generation parameters include:

- **PREFIX_FACTOR** – The threshold for prefix match is defined by a configurable prefix match factor which is an integer PREFIX_MATCH_FACTOR. For example: Nᵢ = “MICRO” and Mᵢ = “MICROSOFT”
  - If Nᵢ matches the beginning of Mᵢ and PREFIX_MATCH_FACTOR* len(Nᵢ) >= len(Mᵢ) then the match is a prefix match.
  - Setting PREFIX_MATCH_FACTOR to 2 means that Nᵢ should be at least half the length of Mᵢ. Setting PREFIX_MATCH_FACTOR to 3 indicates that Nᵢ should be at least one-third the length of Mᵢ.
  - The default value for PREFIX_MATCH_FACTOR is 2.

- **PREFIX_ADJWGT** – If a prefix match is identified, the prefix_adjwgt is used to adjust the prefix token (for example, “MICRO”) weight in the following manner:
  - (Prefix compare token)Nᵢ_wgt = MIN(Nᵢ_wgt, Mᵢ_wgt) - PREFIX_ADJWGT
  - The default value for PREFIX_ADJWGT is 100 or 1.0.

- **PREFIX_MINWGT** – Used as a lower boundary for any prefix weight adjusted token. The prefix adjusted weight never falls below this weight value. The default value for prefix_MINWGT is 50 or .5.

- **PREFIX_MAXWGT** – Used as an upper boundary for any prefix weight adjusted token. The prefix adjusted weight never goes above this weight value. The default value for prefix_MAXWGT is 300 or 3.0.

- **COMPOUND_ADJWGT** – A compound match is detected when comparing tokens “MICRO” “SOFT” vs. “MICROSOFT”. The compound_adjwgt is used to adjust the compound token (“MICRO”) weight in the following manner:
  - (Compound compare token “MICRO”)Nᵢ_wgt = MIN(Nᵢ_wgt, Mᵢ_wgt) - COMPOUND_ADJWGT
  - The default value for compound_adjwgt is 50 or .5.
• **COMPOUND_MINWGT** – Used as a lower boundary for any compound weight adjusted token. The compound adjusted weight never falls below this weight value. The default value for compound_MINWGT is 50 or .5.

• **COMPOUND_MAXWGT** – Used as an upper boundary for any compound weight adjusted token. The compound adjusted weight never goes above this weight value. The default value for compound_MAXWGT is 400 or 4.0.

**Comparison parameters**

The following conditions and penalties are used in comparison to determine the final score.

**Position penalties.** CXNM uses the following to check for position penalties.

- **__CELLEDIFF_MINIDX**
- **__CELLEDIFF_MAXIDX**
- **__CELLEDIFF_ADJWGT %d (d=2,3,4).** “d” refers to default disagreement weight.

CELLEDIFF parameters are applied during the comparison of one token to another. When two tokens that match have a cell difference of 2, the total weight is reduced by CELLEDIFF_ADJWGT_2. The weight is checked to make sure that it is within the range limits.

CELLEDIFF_MAXIDX is used to check that the cell difference does not exceed this Max.

**Edit-distance.** The following parameters are used to check the phone edit-distance. If it is equal to MCCIDX_EQUAL, then it is an exact match, otherwise it is a partial match.

- **__DIST_MINIDX**
- **__DIST_MAXIDX**
- **__DIST_MCCIDX_EQUAL**
- **__DIST_MCCIDX_PARTIAL**

If the edit-distance from comparing phone numbers is equal to DIST_MCCIDX_EQUAL, they are considered an exact match. Similarly, if the distance is equal to DIST_MCCIDX_PARTIAL, they are considered partial match.

**Index calculation.** CXNM uses these parameters to calculate the index for 2dim tables. It also checks for EQUAL or PARTIAL matches (similar to the ones above).

- **__NORM_MINIDX**
- **__NORM_MAXIDX**
- **__NORM_ADJWGT %d (d=2,3,4).** “d” refers to default disagreement weight.
- **__NORM_MCCIDX_EQUAL**
- **__NORM_MCCIDX_PARTIAL**

The NORM_MCCIDX values are used for the address equal or partial match.

NORM_MAXIDX and NORM_MINIDX are used to bind the Normalized value from the Address matching.

NORM_ADJWGT is the final lookup weight for CXNM.
**Partial string matches.** These parameters are used for penalties for the partial string matches:

- __EDITDIST_ADJWGT
- __EDITDIST_FACTOR
- __EDITDIST_MINWGT
- __EDITDIST_MAXWGT
- __INITIAL_ADJWGT
- __INITIAL_MINWGT
- __INITIAL_MAXWGT
- __NICKNAME_ADJWGT
- __NICKNAME_MINWGT
- __NICKNAME_MAXWGT
- __NICKMETA_ADJWGT
- __NICKMETA_MINWGT
- __NICKMETA_MAXWGT
- __PHONETIC_ADJWGT
- __PHONETIC_MINWGT
- __PHONETIC_MAXWGT

**EDITDIST_FACTOR** is used to determine if the two strings can use an edit-distance comparison. **EDITDIST_ADJWGT** is the penalty applied for an edit-distance match. The final weight is then checked to be within the ranges by using the MIN and MAX values.

**Acronym matches.** The following use the MIN and MAX values to check the edit-distance in acronyms. If the weights are less than MIN or greater than MAX, they are adjusted back to the MIN and MAX.

- __ACRONYM_MINLEN
- __ACRONYM_MAXLEN

**CXNM example:**

Consider comparing the strings BILL JOHNSONS TRUCKS and B JOHNSONS BIG TRUCKS. The following weights are the exact match weights and some penalties that are used:

**Bill**
- 30

**Johnsons**
- 50

**Trucks**
- 40

**B**
- 20

**Big**
- 30

**Initial Adjustment Weight**
- 5

**Position Adjustment Weight**
- 5
The comparison is done in this series of steps. At every step, the best weight is carried forward.

1. B with [BILL, JOHNSONS, TRUCKS]
   - B and BILL partially match and the weight is:
     20-5=15 (exact match – initial adjustment weight)

2. JOHNSONS with [BILL, JOHNSONS AND TRUCKS]
   - JOHNSONS and JOHNNSONS match exactly and the weight is 50+15=65 (exact match + previous weight).

3. BIG with [BILL, JOHNSONS AND TRUCKS]
   - No matches

4. TRUCKS with [BILL, JOHNNSONS, TRUCKS]
   - TRUCKS and TRUCKS match exactly and the weight is: 65+40-5=100
     (previous weight + exact match weight – position penalty)

The position penalty weight is applied in this case, since there was a positional gap between this match and the previous match.

Final Score:
• The total for B JOHNNSONS BIG TRUCKS is 20+50+30+40=140
• The total for BILL JOHNNSONS TRUCKS is 30+50+40=120
• The average is 140+120=260/2=130
• The normalized score is now: (100/130*15) =12.30 (this is done since the scale is from 0 -15). This is now looked up in the mpi_wgtsval table under _NORM_ADJWGT_12 and that is the score used.

**CXNM_CS**
The CXNM_CS (CS indicates “context sensitive”) comparison functions enables business name token weights to reflect locality.

For example, within the local of the Phoenix metropolitan area, the token “Phoenix” has a high frequency and a low weight. Outside of this local, the frequency is low and weight high. Similarly, if Bell Flower Shop is located on Bell Road, the token “Bell” should have a low value.

CXNM_CS uses the cmpargs argument to specify the type of phonetic function to be applied.

**Number of roles**
2

**Weight table**

mpi_wgtsval, mpi_wgthead

The CXNM_CS functions allow you to set the address location of the business in cmprole2, which is used to determine the presence of any address tokens in the business name, and adjust the weight of the token appropriately.

**CXNM_CS example:**

Bell Flower Shop, 7611 E. Bell Road, Phoenix, AZ
Bell Pizza, 7621 E. Bell Road, Phoenix, AZ

**PXNM (comparison function)**
The PXNM function is used to compare personal names.
Like other XNM functions, PXNM uses cmpargs to specify the type of metaphone algorithm to be used.

PXNM works in the same way as QXNM, but does not use a maximum weight parameter. If the words are in the same place in both comparison strings, the weight is bumped up by a factor of 20. If they are off by one position, it is bumped up by 10. If not, the weight is retained as is. The name weight is then the sum of the individual weight token comparisons.

Another difference is that PXNM does not check for edit-distance matches. If there are any suffixes, they are compared in the same manner as in QXNM and the corresponding weight is added to the previous weights to get the final weight.

**QXNM (comparison function)**
The QXNM routine is used for name comparisons.

Like other XNM functions, QXNM uses cmpargs to specify the type of phonetic function to be applied.

**Number of roles**
1

**Weight table**
mpi_wgtsval, mpi_wgtold

**Other tables**
mpi_cmpspec, mpi_strequi(NICKNAME), mpi_cmphead

The steps for QXNM are similar to CXNM, with slight variations.
1. Compare each token in the first string with the each token in the second string.
2. Find the pairings that give the highest score, not using a token more than once. If an exact match is found, that token is not used in the comparison thereafter. If the two strings have different lengths, the extra tokens in the longer string will go unused.
3. Add up the positive scores for the matches including position bonuses. If the total positive score exceeds the maximum total weight, (FULLNAMEMAXWGT), the positive score is recomputed as maximum total plus positional bonus.
4. Subtract the token disagreement scores.
5. You have the option to specify RealNames in mpi_strword using a strcode of QXNM-NAME. The words specified in this table are not compared for phonetic or edit-distance matches or nicknames of phonetics. Nickname comparison can be included or excluded depending on the cmpargs value (which can be NICK or NONICK). All real names need to exist in mpi_strword with a strcode of QXNM-NAME. Baseline real name values ship with the Master Data Engine in the hub_install_path\sql\mpi_cmpqxnm.sql file. You can load this file by hand or via the madsqsql utility to populate the real name values in mpi_strword. See “Enabling RealNames with QXNM” on page 137 for instructions on adding the QXNM-NAME strcode to your IBM Initiate Workbench project.

Note that for real names to function, both names that are being matched should be in the real names table.
**Position bonus**

If the two tokens match exactly and are in the same position in the string, the weights are bumped up by adding 20. If they are off by one position, then they are bumped up by adding 10.

For example, JOHN:TIMBER:SMITH and JOHN:TIMBER:SMITH get a bump of 20 for each token, thus making it a total of 60.

JOHN:SMITH:TIMBER and JOHN:TIMBER:SMITH get a bump of 10 for SMITH and TIMBER and 20 for JOHN, thus the total weight is 20+10+10=40.

QXNM example:

QXNM: JONES | BOB | R | | -- JONES | ROBERT | J | | , wgt= +2.94, mcc=D

**EXACT MATCH WEIGHTS**

<table>
<thead>
<tr>
<th>Word</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>JONES</td>
<td>2.58</td>
</tr>
<tr>
<td>BOB</td>
<td>3.55</td>
</tr>
<tr>
<td>ROBERT</td>
<td>1.87</td>
</tr>
<tr>
<td>R</td>
<td>1.68</td>
</tr>
<tr>
<td>J</td>
<td>1.47</td>
</tr>
</tbody>
</table>

Nickname Penalty = 1.23
Initial Penalty = 2.22

This scenario illustrates all pair comparisons for Bob Jones and Robert Jones.

Start with JONES in first list. Since the first compare results in an exact match, stop here. Neither JONES token is compared again.

Now, compare BOB to the remaining tokens and R to the remaining token.

**QXNM Detail:**

QXNM[1]: JONES -- JONES, wgt= +2.68, Exact Match
QXNM[2]: BOB -- ROBERT, wgt= +0.64, Nickname Match
QXNM[3]: BOB -- J, wgt= -0.58, Mismatch
QXNM[4]: R -- ROBERT, wgt= +0.60, Initial Match
QXNM[5]: R -- J, wgt= -0.58, Mismatch

Now order the list by weights.

QXNM[1]: JONES -- JONES, wgt= +2.68, Exact Match
QXNM[2]: BOB -- ROBERT, wgt= +0.64, Nickname Match
QXNM[4]: R -- ROBERT, wgt= +0.60, Initial Match
QXNM[3]: BOB -- J, wgt= -0.58, Mismatch
QXNM[5]: R -- J, wgt= -0.58, Mismatch

The first pair in the list is JONES-JONES, so this pair is our first match.

The second pair is BOB-ROBERT. The function checks to see that neither token has been used yet. Since they have not, the detail is:

QXNM[1]: JONES -- JONES, wgt= +2.68, Exact Match
QXNM[2]: BOB -- ROBERT, wgt= +0.64, Nickname Match
The next pair is R-ROBERT. However, since ROBERT has already been matched up we are going to skip that pair.

Next are BOB and J. Again, BOB has been used. Next is R – J, neither of which has been used. Therefore, our final pairings are:

QXNM[1]: JONES -- JONES, wgt= +2.68, Exact Match
QXNM[2]: BOB -- ROBERT, wgt= +0.64, Nickname Match
QXNM[5]: R -- J, wgt= -0.58, Mismatch

Tip: If you get through the ordered list and there are one or more tokens in one of the lists that do not appear in your pairings, then that is fine. That token does not contribute to the scoring. There are never tokens in both lists that do not appear.

**QXNM parameters**

There are two types of parameters used by XNM functions. The first is a set of parameters used during the comparison step of weight generation. The second set of parameters will use the outputs from weight generation and are used during the actual data comparison to determine a final score.

**Weight generation comparison parameters:**

With normal edit distance processing, there are names that might not be recognized. Using the following weight generation parameters for prefix and compound address matching can significantly improve comparisons.

Weight generation parameters include:

- **PREFIX_FACTOR** – The threshold for prefix match is defined by a configurable prefix match factor which is an integer PREFIX_MATCH_FACTOR. For example: 
  \[ N_i = \text{“MICRO”} \quad \text{and} \quad M_j = \text{“MICROSOFT”} \]
  If \( N_i \) matches the beginning of \( M_j \) and \( \text{PREFIX_MATCH_FACTOR} \times \text{len}(N_i) \geq \text{len}(M_j) \) then the match is a prefix match.
  Setting \( \text{PREFIX_MATCH_FACTOR} \) to 2 means that \( N_i \) should be at least half the length of \( M_j \). Setting \( \text{PREFIX_MATCH_FACTOR} \) to 3 indicates that \( N_i \) should be at least one-third the length of \( M_j \).
  The default value for \( \text{PREFIX_MATCH_FACTOR} \) is 2.

- **PREFIX_ADJWGT** – If a prefix match is identified, the prefix_adjwgt is used to adjust the prefix token (for example, “MICRO”) weight in the following manner:
  \[ (\text{Prefix compare token})N_i \text{-wgt} = \text{MIN}(N_i \text{-wgt}, M_j \text{-wgt}) - \text{PREFIX_ADJWGT} \]
  The default value for \( \text{PREFIX_ADJWGT} \) is 100 or 1.0.

- **PREFIX_MINWGT** – Used as a lower boundary for any prefix weight adjusted token. The prefix adjusted weight never falls below this weight value. The default value for \( \text{PREFIX_MINWGT} \) is 50 or .5.

- **PREFIX_MAXWGT** – Used as an upper boundary for any prefix weight adjusted token. The prefix adjusted weight never goes above this weight value. The default value for \( \text{PREFIX_MAXWGT} \) is 300 or 3.0.

- **COMPOUND_ADJWGT** – A compound match is detected when comparing tokens “MICRO” “SOFT” vs. “MICROSOFT”. The compound_adjwgt is used to adjust the compound token (“MICRO”) weight in the following manner:
  \[ (\text{Compound compare token “MICRO”)N_i \text{-wgt} = \text{MIN}(N_i \text{-wgt}, M_j \text{-wgt}) - \text{COMPOUND_ADJWGT} \]
  The default value for \( \text{COMPOUND_ADJWGT} \) is 50 or .5.
• **COMPOUND_MINWGT** – Used as a lower boundary for any compound weight adjusted token. The compound adjusted weight never falls below this weight value. The default value for compound_MINWGT is 50 or .5.

• **COMPOUND_MAXWGT** – Used as an upper boundary for any compound weight adjusted token. The compound adjusted weight never goes above this weight value. The default value for compound_MAXWGT is 400 or 4.0.

**Comparison parameters**

The following conditions and penalties are used in comparison to determine the final score.

**Partial string matches.** These parameters are used for penalties for the partial string matches.

- __EDITDIST_ADJWGT
- __EDITDIST_FACTOR
- __EDITDIST_MINWGT
- __EDITDIST_MAXWGT
- __FULLNAME_MAXWGT
- __INITIAL_ADJWGT
- __INITIAL_MINWGT
- __INITIAL_MAXWGT
- __NICKNAME_ADJWGT
- __NICKNAME_MINWGT
- __NICKNAME_MAXWGT
- __NICKMETA_ADJWGT
- __NICKMETA_MINWGT
- __NICKMETA_MAXWGT
- __PHONETIC_ADJWGT
- __PHONETIC_MINWGT
- __PHONETIC_MAXWGT

EDITDIST_FACTOR is used to determine if the two strings can use an edit-distance comparison. EDITDIST_ADJWGT is the penalty applied for an edit-distance match. The final weight is then checked to be within the ranges by using the MIN and MAX values.

**Bucketing**

The creation of bucket values is a two-step process.

First a bucketing function is applied to the compare value that was generated by the standardization function.

The second step is the generation of bucket values. A bucket generation function (specified in mpi_dvybkt.genfunccode) takes the output of the bucket function and modifies it to produce the bucket values. The bucket generation function formats the “name” of the bucket. “Name” is used within quotation marks because the bucket name is converted to a hash number that represents a real value.

The maximum number of bucket roles (bktroles) is 15.
Bucket functions

Bucketing functions enable flexibility for bucketing routines and assist in candidate selection by identifying groups of shared information.

There are bucket functions for addresses, attributes, and dates.

**ADDR2**
This bucketing function generates a list of addresses.

Use this function when the standardization is processed by one of the "ADDR2" standardization functions.

**ATTR (bucket function)**
The ATTR bucket function generates a single value.

While other bucketing functions are designed to work with specified standardization output, ATTR can be used with most attributes. When ATTR is used, the compare value is sent unfiltered to bucket generation.

**BXNM (bucket function)**
The BXNM bucket function creates a list of name tokens from the business name compare string.

Use this bucket function when the standardization function is BXNM.

**CDATE (bucket function)**
The CDATE bucketing function creates a list of date tokens.

Use the CDATE bucket function when your standardization function is CDATE. Likewise, use the CDATE bucket generation function with this bucket function.

**CXNM (bucket function)**
The CXNM bucket function creates a list of name tokens from the business comparison string.

Use this bucket function when the standardization function is CXNM.

**DATE (bucket function)**
The DATE bucket function generates a single date value.

When the DATE bucket function is used, the comparison data is sent unfiltered to bucket generation. Use this bucket function when the standardization function is DATE or GRDATE and the generation function is one of the following: ASIS, DTY4SMD, DTY4MM, DYMMDD. If you want to apply one of the other generation functions to a date, you must use the bucket function ATTR.

**PXNM (bucket function)**
The PXNM bucket function creates a list of name tokens from the personal name compare string.

Use this function when the standardization function is PXNM or QXNM.

Generation Functions

Generation function codes are specified in mpi_dvdybkt and determine what further processing is applied to the output of the generation functions.
**ASIS**

ASIS generation function does not modify the input value, rather it leaves each token “as is.”

ASIS can be used with any attribute or bucket function.

**CDATE**

The CDATE bucket generation function takes the standardized date token (the date value plus and minus the resolution value) from the CDATE standardization function and generates buckets for dates in the low date range to the high date range.

This function is used with the CDATE standardization, comparison, and bucket functions.

The CDATE bucket generation function is based upon a separate quantum value, called delbkt, which is an integer number of years. The delbkt value by default is 1. It can also be input through bktArgs (dvdArgs in mpi_dvdynkt).

For a date, the year and the resolution value are used to create the base-bucket value (bbv). For example, a year date plus/minus the resolution yields a year date range. A missing month in the date is substituted with 07 and a missing day is substituted with 14.

Buckets are then created for dates in the range of low_bbv/delbkt to hight_bbv/delbkt. For example, you have a date/resolution standardized value of 1998.01.01.Y1. Using a delbkt value of 1, generates a low base-bucket value of 1997 and a high base-bucket value of 1999. Thus, buckets are created in the low/high range of 1997 to 1999.

**DTMMDD**

The DTMMDD generation function generates a date with a two-digit month and two-digit date.

This function uses only the MMDD portion of the date token to group dates by month and day. Year is ignored.

**DTY4NM**

The DTY4NM generation function generates a date with four-digit year and month.

This function uses only the YYYYMM portion of the date token to group dates by month and year.

**DTY4SMD**

The DTY4SMD generation function generates a date with a four-digit year followed by a sorted month and date (for example, 19711205 produces 19710512).

This function applies canonical date transform to each token (YYYY sorted MM DD) and each date gets a bucket in YYYYMMDD format.

**EQUI**

The EQUI generation function uses a nickname table to generate a list of equivalences.

The nickname table maps y name value to x name value.
EQMETA
This generation function combines the functions of EQUI and META.

A nickname table is used to generate a list of equivalences and then a phonetic algorithm is applied to these values. You can specify derivation arguments to the function that indicate the phonetic function to use, similar to the comparison functions.

GEO (bucket function)
The GEO generation function generates a location value based on latitude/longitude coordinates.

The input for the GEO generation function must come from an attribute standardized by the GEO standardization function. See the standardization function "GEO (standardization function)" on page 271 for the details of the input and standard formats. The GEO generation function can be used only with the ATTR bucketing function.

Dvdargs are:
- Precision - specified as a number of decimal places
- Neighbor generation - specified as NEIGHBORS

The GEO generation function operates by reducing the resolution of location coordinates so that locations that are close together will generate the same bucket values. The resolution is controlled by specifying the number of fractional digits to preserve in the coordinates (fractional degrees). Input values are truncated to this number of digits by rounding. The default resolution is two decimal digits. The resolution can be changed by providing an argument with the number of digits wanted.

This approach creates a “grid” with the given resolution. Since some locations might be close to the edges of this grid, it can be desirable to generate multiple bucket values for a single location that include neighboring grid locations. This is accomplished by generating bucket values not only for the original location, but also for the locations that are plus or minus one resolution unit in both latitude and longitude. For example, the location N30.301 W97.697 would generate the following bucket values:

N30.30 W97.70
N30.31 W97.70
N30.29 W97.70
N30.30 W97.71
N30.30 W97.69

By default, neighboring buckets are not created. To enable generation of neighboring buckets, include a dvd argument named “NEIGHBORS”.

GNRMETA
GNRMETA bucket generation function can be used for data sets containing names.
If your data set contains multicultural names, this function can be especially beneficial. In the IBM Initiate Workbench algorithm editor, GNRMETA is identified as GNR & Phonetic.

GNRMETA calls out to IBM InfoSphere® Global Name Recognition (GNR) for name variants and corresponding percentages produced by the GNR analyze() method. The percentages are the frequencies of a particular variant in comparison to other variants. The variants are then filtered by using a percentage threshold setting. Only those variants that are greater than that percent are used in bucketing. The anonymous value (ANON) handling is done before the input values are sent to GNR.

GNRMETA is similar to using EQMETA with an equivalency string code (equristrcode) of NICKNAME. EQMETA, with NICKNAME, looks up the various nickname forms of a token and then passes it through the META function. With GNRMETA, the lookup is done with GNR instead of a NICKNAME table.

There are two dvddArgs used with GNRMETA. The first is the phonetic function and the second is the percentage threshold value, which is specified as percent=value. The value must be an integer. For example, percent=10.

GNRMETA can be used with any of the existing IBM Initiate Master Data Service standardization, comparison, and phonetic functions. You must use either PXNM or BXNM bucketing functions.

**META [1,2,3,4]**

There are four META generation functions; META1, META2, META3, and META 4.

These functions apply a phonetic transformation to inputs. See also METAPHONE, PREFIXMAP, IDENTAPHONE, and ARABPHONE functions.

**METAPHONE, PREFIXMAP, IDENTAPHONE, ARABPHONE, FRPHONE**

These functions apply a phonetic transformation to input.

METAPHONE is used to index words by sound. (also be called META1).

PREFIXMAP (also called META2) uses a truncation mapping such that for each token, the truncation function outputs the first three characters. If the token is less than three characters, the entire token is output.

IDENTAPHONE (also called META3) uses an IBM proprietary phonetic algorithm.

ARABPHONE (also called META4) is used to apply phonetics to the English translation of Arabic names. (This function does not work on Arabic character sets; see "ARABICPHONE").

FRPHONE applies an IBM proprietary transformation like IDENTAPHONE, but is suited to predominantly French names.

**ARABICPHONE**

This phonetic function is designed to handle common Arabic homophones.

Use the ARABICPHONE function in your algorithm configuration with the RXNM standardization function, PXNM bucketing function, and QXNM comparison function. You can use any of the bucket generation functions.
NGRAM

Use the NGRAM generation function to help improve candidate selection when your attributes contain similar values, but might vary in spelling or position.

The NGRAM bucket generation function can be used with the supporting bucketing functions of ATTR, BXNM, CXNM, and PXNM.

An NGRAM is a sub-sequence of size N generated from an input sequence. Collectively, NGRAMS are all the sub-sequences of size N that can be generated from the input sequence. This function creates buckets based on the N-grams generated from each data item.

The value, or size, of “N” is user-defined and set as a dvdArg in mpi_dvdybkt. If not specified, the default value of 3 is used. For names, a gram size of 3 or 4 is suggested. For phone numbers, a gram size of 7 is suggested.

An example of NGRAM output is:

N = 3 with the input of “MOHAMMED” results in MOH OHA HAM AMM MME MED

N = 4 with the input of “6345111” results in 6345 3451 4511 5111

Two additional dvdargs, TINYGRAMS and REMOVEVOWELS, are available to further break down the output and enhance bucket results. Multiple dvdargs can be specified by using either a comma (,) or plus (+) as separators.

- TINYGRAMS
  The TINYGRAMS dvdArg generates grams from tokens with a length less than N. The NGRAMS default dvdArg setting does not use TINYGRAMS and generate buckets only from tokens with the length equal to the size of N. Use the TINYGRAMS option when name tokens exist that are smaller than your choice of N, but might be significant for bucketing purposes and would otherwise be ignored.
  
  For example, use the name AL AS’AD BEN HANI. Without TINYGRAMS, the output would be:
  ASA ASD BEN HAN ANI
  With TINYGRAMS, the output is:
  AL ASA ASD BEN HAN ANI
  By not using TINYGRAMS, the token ‘AL’ would be ignored in bucketing. In many cases, the presence of this token could be significant to the comparison results.

- REMOVEVOWELS
  This function removes all occurrences of A, E, I, O, and U from medial and final token positions. The default is to keep all vowels. Use this option to increase the chance that similar names differing by their vowels will bucket.
  For example, use the name MOHAMMAD HASSAN AKHUND. Without REMOVEVOWELS, the output is:
  MOH OHA HAM AMM MMA MAD HAS SSA SAN AKH KHU HUN UND
  With REMOVEVOWELS, the output is:
  MHM HMM MMD HSS SSN AKH KHN HND
The NGRAM function can generate a large number of buckets and should be used with Frequency-Based-Bucketing.

**NRANGE**
The NRANGE function generates numeric range buckets and is used for integer data only.

The range buckets are normally used when you want buckets that link members who are “close” (for example, height, weight). This function is especially helpful when you have only an estimation of these attributes.

When you specify NRANGE in mpi_dvdybkt, you must also specify a bucket string range code (rbktstrcode), which has an entry in mpi_strhead (strcode field). There is a separate table mpi_strnbkt that holds these range values.

Example of ranges:

W89 - for weights between 80-89
W99 - for weights between 90-99
...

For each weight (or attribute value), three bucket values are generated:

- the range itself,
- the previous range, and
- the subsequent range.

For the values less than the minimum or greater than maximum, the min and max are used.

**SRANGE**
The SRANGE generation function is used to bucket strings with range values that compare close to each other.

An example of when to use this function would be when you have both birth years and birth dates and want to bucket them together. In such a situation, NRANGE does not work since, for example, 19739523 and 1973 are not “close” as integers. However, the values in the strings are close.

When you specify SRANGE in mpi_dvdybkt, you must also specify a bucket string range code (rbktstrcode), which has an entry in mpi_strhead (strcode field). The mpi_strsbkt is used instead of mpi_strnbkt.

Example:

BY_1900 for strings between 1900 and 1909
BY_1910 for strings between 1910 and 1919
BY_1919 for strings between 1920 and 1929

For both NRANGE and SRANGE values, it would be best to have more than three range values. With only three values, all values might possibly be in the same bucket.

**SORTED**
The SORTED generation function sorts the characters of each token and transforms them for bucketing.
Sorting gives more room to bucket more members.
Appendix B. Algorithm validation rule syntax

If a new standardization function, comparison function, bucketing function, or generation function is added to the Initiate database, it is recommended that an associated validation rule be added to the workbench install path\plugins\com.initiatesystems.workbench_9.7.x\rules\algorithm-rules.xml file. Without a validation rule defined for the new function, the IBM Initiate Workbench algorithm editor cannot detect validation problems associated with its use within an algorithm. You must add the validation rule using the syntax defined in the following sections.

Adding validation rules

The validation rules are added to the algorithm-rules.xml using a text or XML editor.

1. Open the workbench install path\plugins\com.initiatesystems.workbench_9.7.x\rules\algorithm-rules.xml file using an appropriate editor.
2. Locate the section where your new rule will best fit. You can create rules for the following function types:
   • Standardization functions
   • Comparison functions
   • Bucketing functions
   • Generation functions

   A sample of the validation-rules.xml is included in "Sample Validation Rules XML" on page 337.

   Note: Any rules more than 255 characters in length must be split into multiple properties, numbered consecutively by Record Number.
3. Refer to the following sections for details on defining your validation rules.

Standardization function rules

stdFunction describes a validation rule for a standardization function. We use XML format for configuring these, for example:

<stdFunction name="ATTR" attrTypes="ANY" anonStrCode="Y" equiStrCode="Y" output="SIMPLE"><fldArgs allowed="ANY" maxFields="1"/></stdFunction>

using the following variables:

• name - the name of the standardization function; see mpi_stdfunc.stdfunccode
• attrTypes - list of allowed segattr types, delimited by "~"; "ANY" means all segattr types are allowed; see mpi_segattr.segcode
• anonStrCode - indicates if an anonstrcode is used for filtering; either "Y" or "N"
• equiStrCode - indicates if an equistrcode is used; either "Y" or "N"
• output - (future use) indicates type of output, either "SIMPLE" (single value) or "STRUCTURED" (list of values)
• fldArgs - 0, 1, or more arguments passed to the standardization function; the IBM Initiate Workbench algorithm editor checks for duplicate fields in fldArgs list
allowed - specifies the expected fldArg values; lists are comma-delimited; options are delimited by "~"; "ANY" means any single DB field allowed; if ANY is specified and the standardization function is connected to an attribute, then the fldArgs property (in the Properties view) is compared against the list of valid field values for the attribute's segcode from mpi_segxfld

- minFields - the minimum number of fldArgs required
- maxFields - the maximum number of fldArgs allowed

### Comparison function rules

cmpFunction describes a validation rule for a comparison function. We use XML format for configuring these, for example:

```xml
<cmpFunction name="DR1D1A"><input role="1" allowed="SIMPLE"/></cmpFunction>
```

using the following variables:

- name - the name of the comparison function; see mpi_cmpfunc
- input - identifies a single comparison role input
- role - the role identifier, numbered consecutively from "1" to "4"
- allowed - (future use) indicates type of data allowed as input, either "SIMPLE" (single value) or "STRUCTURED" (list of values)

### Bucketing Functions

bktFunction describes a validation rule for a bucketing function. We use XML format for configuring these, for example:

```xml
<bktFunction name="DATE" allowed="DATE1~GRDATE"
genFunctions="ASIS~DTY4SMD~DTY4MM~DYMMDD"></bktFunction>
```

using the following variables:

- name - the name of the bucketing function; see mpi_bktfunc
- allowed - standardization functions allowed as input, delimited by "~"
- genFunctions - generation functions required, delimited by "~"; see mpi_genfunc

### Generation Functions

genFunction describes a validation rule for a generation function. We use XML format for configuring these, for example:

```xml
<genFunction name="ASIS" allowed="ANY" equiStrCode="N"
rbktStrCode="N"></genFunction>
```

using the following variables:

- name - the name of the generation function; see mpi_genfunc
- allowed - the bucketing functions allowed, delimited by "~"; "ANY" means all bucketing functions are allowed
- exclude - list of bucketing function exclusions to the "allowed" list, delimited by "~"
- equiStrCode - indicates if an equistrcode is required; either "Y" or "N"
- rbktStrCode - indicates if an rbktstrcode is required; either "Y" or "N"
Sample Validation Rules XML

The following sample XML shows the validation rule syntax as well as the structure of the complete XML document after the IBM Initiate Workbench algorithm editor retrieves it from the Initiate database:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<algorithm-rules version="1.0">
  <stdFunctions>
    <stdFunction name="ATTR" attrTypes="ANY" equiStrCode="Y" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
      <fldArgs maxFields="1" allowed="ANY" />
    </stdFunction>
    <stdFunction name="ATTRA" attrTypes="ANY" equiStrCode="Y" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
      <fldArgs maxFields="1" allowed="ANY" />
    </stdFunction>
    <stdFunction name="ATTRN" attrTypes="ANY" equiStrCode="Y" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
      <fldArgs maxFields="1" allowed="ANY" />
    </stdFunction>
    <stdFunction name="EMAIL" attrTypes="MEMATTR" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
      <fldArgs allowed="attrval" />
    </stdFunction>
    <stdFunction name="PXNM" attrTypes="MEMNAME" equiStrCode="N" output="STRUCTURED" anonStrCode="Y" cmapStrCode="Y">
      <fldArgs minFields="1" maxFields="6" allowed="onmlast,onmfirst,onmmiddle,onmprefix,onmsuffix,onmdegree~onmlast,onmfirst,onmmiddle,onmprefix,onmsuffix,onmtitle" />
    </stdFunction>
    <stdFunction name="BXNM" attrTypes="MEMNAME~MEMATTR" equiStrCode="N" output="STRUCTURED" anonStrCode="Y" cmapStrCode="Y">
      <fldArgs maxFields="6" allowed="ANY" />
    </stdFunction>
    <stdFunction name="CXNM" attrTypes="MEMNAME~MEMATTR" equiStrCode="N" output="STRUCTURED" anonStrCode="Y" cmapStrCode="Y">
      <fldArgs maxFields="1" allowed="ANY" />
    </stdFunction>
    <stdFunction name="USADDR" attrTypes="MEMADDR" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="Y">
      <fldArgs minFields="1" maxFields="4" allowed="stline1,stline2,stline3,stline4" />
    </stdFunction>
    <stdFunction name="CNADDR" attrTypes="MEMADDR" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="Y">
      <fldArgs minFields="1" maxFields="4" allowed="stline1,stline2,stline3,stline4" />
    </stdFunction>
    <stdFunction name="USADDR2" attrTypes="MEMADDR" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="Y">
      <fldArgs minFields="1" maxFields="7" allowed="stline1,stline2,stline3,stline4,city,state,zipcode" />
    </stdFunction>
    <stdFunction name="CNADDR2" attrTypes="MEMADDR" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="Y">
      <fldArgs minFields="1" maxFields="7" allowed="stline1,stline2,stline3,stline4,city,state,zipcode" />
    </stdFunction>
    <stdFunction name="NAADDR2" attrTypes="MEMADDR" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="Y">
      <fldArgs minFields="1" maxFields="7" allowed="stline1,stline2,stline3,stline4,city,state,zipcode" />
    </stdFunction>
    <stdFunction name="UKADDR2" attrTypes="MEMADDR" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="Y">
      <fldArgs minFields="1" maxFields="7" allowed="stline1,stline2,stline3,stline4,city,state,zipcode" />
    </stdFunction>
  </stdFunctions>
</algorithm-rules>
```
<stdFunction name="INTADD2" attrTypes="MEMADDR" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="Y">
  <fldArgs minFields="1" maxFields="7" allowed="stline1,stline2,stline3,stline4,city,state,zipcode" />
</stdFunction>

- <stdFunction name="USZIP" attrTypes="MEMADDR" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
  <fldArgs allowed="zipcode" />
</stdFunction>

- <stdFunction name="CNZIP" attrTypes="MEMADDR" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
  <fldArgs allowed="zipcode" />
</stdFunction>

- <stdFunction name="NAZIP" attrTypes="MEMADDR" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
  <fldArgs allowed="zipcode" />
</stdFunction>

- <stdFunction name="UKZIP" attrTypes="MEMADDR" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
  <fldArgs allowed="zipcode" />
</stdFunction>

- <stdFunction name="INTZIP" attrTypes="MEMADDR" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
  <fldArgs allowed="zipcode" />
</stdFunction>

- <stdFunction name="PHONEEND" attrTypes="MEMPHONE" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
  <fldArgs allowed="phnumber" />
</stdFunction>

- <stdFunction name="PHONE1" attrTypes="MEMPHONE" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
  <fldArgs allowed="phnumber" />
</stdFunction>

- <stdFunction name="PHONE2" attrTypes="MEMPHONE" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
  <fldArgs minFields="1" maxFields="3" allowed="pharea,phnumber,phextn" />
</stdFunction>

- <stdFunction name="INTPHONE1" attrTypes="MEMPHONE" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
  <fldArgs minFields="1" maxFields="3" allowed="pharea,phnumber,phextn" />
</stdFunction>

- <stdFunction name="INTPHONE2" attrTypes="MEMPHONE" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
  <fldArgs minFields="1" maxFields="3" allowed="pharea,phnumber,phextn" />
</stdFunction>

- <stdFunction name="AUSTPH" attrTypes="MEMPHONE" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
  <fldArgs allowed="phnumber" />
</stdFunction>

- <stdFunction name="AUSTPOST" attrTypes="MEMADDR" equiStrCode="N" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
  <fldArgs allowed="zipcode" />
</stdFunction>

- <stdFunction name="IDENT1" attrTypes="MEMIDENT" equiStrCode="Y" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
  <fldArgs allowed="idnumber" />
</stdFunction>

- <stdFunction name="IDENT1A" attrTypes="MEMIDENT" equiStrCode="Y" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
  <fldArgs allowed="idnumber" />
</stdFunction>

- <stdFunction name="IDENT1N" attrTypes="MEMIDENT" equiStrCode="Y" output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
  <fldArgs allowed="idnumber" />
</stdFunction>

338 Workbench User's Guide
Appendix B. Algorithm validation rule syntax

```xml
<function>
  <name>IDENT2</name>
  <attrTypes>MEMIDENT</attrTypes>
  <equiStrCode>Y</equiStrCode>
  <output>SIMPLE</output>
  <anonStrCode>Y</anonStrCode>
  <cmapStrCode>N</cmapStrCode>
  <fldArgs allowed="idnumber"/>
</function>

<function>
  <name>IDENT2A</name>
  <attrTypes>MEMIDENT</attrTypes>
  <equiStrCode>Y</equiStrCode>
  <output>SIMPLE</output>
  <anonStrCode>Y</anonStrCode>
  <cmapStrCode>N</cmapStrCode>
  <fldArgs allowed="idnumber"/>
</function>

<function>
  <name>IDENT2N</name>
  <attrTypes>MEMIDENT</attrTypes>
  <equiStrCode>Y</equiStrCode>
  <output>SIMPLE</output>
  <anonStrCode>Y</anonStrCode>
  <cmapStrCode>N</cmapStrCode>
  <fldArgs allowed="idnumber"/>
</function>

<function>
  <name>DATE1</name>
  <attrTypes>MEMDATE</attrTypes>
  <equiStrCode>N</equiStrCode>
  <output>SIMPLE</output>
  <anonStrCode>Y</anonStrCode>
  <cmapStrCode>N</cmapStrCode>
  <fldArgs allowed="dateval"/>
</function>

<function>
  <name>ABSCODE</name>
  <attrTypes>MEMATTR</attrTypes>
  <equiStrCode>N</equiStrCode>
  <output>STRUCTURED</output>
  <anonStrCode>N</anonStrCode>
  <cmapStrCode>N</cmapStrCode>
  <fldArgs allowed="attrval"/>
</function>

<function>
  <name>CJKCXNM</name>
  <attrTypes>MEMNAME~MEMATTR</attrTypes>
  <equiStrCode>N</equiStrCode>
  <output>STRUCTURED</output>
  <anonStrCode>Y</anonStrCode>
  <cmapStrCode>Y</cmapStrCode>
  <fldArgs maxFields="1" allowed="ANY"/>
</function>

<function>
  <name>HAIRCOLOR</name>
  <attrTypes>ANY</attrTypes>
  <equiStrCode>Y</equiStrCode>
  <output>SIMPLE</output>
  <anonStrCode>Y</anonStrCode>
  <cmapStrCode>N</cmapStrCode>
  <fldArgs maxFields="1" allowed="ANY"/>
</function>

<function>
  <name>EYECOLOR</name>
  <attrTypes>ANY</attrTypes>
  <equiStrCode>Y</equiStrCode>
  <output>SIMPLE</output>
  <anonStrCode>Y</anonStrCode>
  <cmapStrCode>N</cmapStrCode>
  <fldArgs maxFields="1" allowed="ANY"/>
</function>

<function>
  <name>GEO</name>
  <attrTypes>MEMATTR~MEMADDR</attrTypes>
  <equiStrCode>N</equiStrCode>
  <output>SIMPLE</output>
  <anonStrCode>N</anonStrCode>
  <cmapStrCode>N</cmapStrCode>
  <fldArgs maxFields="1" allowed="ANY"/>
</function>

<function>
  <name>RACE</name>
  <attrTypes>ANY</attrTypes>
  <equiStrCode>Y</equiStrCode>
  <output>SIMPLE</output>
  <anonStrCode>Y</anonStrCode>
  <cmapStrCode>N</cmapStrCode>
  <fldArgs maxFields="1" allowed="ANY"/>
</function>

<function>
  <name>HEIGHT</name>
  <attrTypes>MEMATTR</attrTypes>
  <equiStrCode>N</equiStrCode>
  <output>SIMPLE</output>
  <anonStrCode>N</anonStrCode>
  <cmapStrCode>N</cmapStrCode>
  <fldArgs allowed="attrval"/>
</function>

<function>
  <name>WEIGHT</name>
  <attrTypes>MEMATTR</attrTypes>
  <equiStrCode>N</equiStrCode>
  <output>SIMPLE</output>
  <anonStrCode>N</anonStrCode>
  <cmapStrCode>N</cmapStrCode>
  <fldArgs allowed="attrval"/>
</function>

<function>
  <name>AGE</name>
  <attrTypes>MEMATTR</attrTypes>
  <equiStrCode>N</equiStrCode>
  <output>SIMPLE</output>
  <anonStrCode>N</anonStrCode>
  <cmapStrCode>N</cmapStrCode>
  <fldArgs allowed="attrval"/>
</function>

<function>
  <name>GRDATE</name>
  <attrTypes>MEMDATE</attrTypes>
  <equiStrCode>N</equiStrCode>
  <output>SIMPLE</output>
  <anonStrCode>Y</anonStrCode>
  <cmapStrCode>N</cmapStrCode>
  <fldArgs allowed="dateval"/>
</function>

<function>
  <name>DATE2</name>
  <attrTypes>MEMATTR~MEMDATE</attrTypes>
  <equiStrCode>Y</equiStrCode>
  <output>SIMPLE</output>
  <anonStrCode>Y</anonStrCode>
  <cmapStrCode>N</cmapStrCode>
  <fldArgs maxFields="1" allowed="ANY"/>
</function>

<function>
  <name>UCSFREQXNM</name>
  <attrTypes>MEMNAME~MEMATTR</attrTypes>
  <equiStrCode>N</equiStrCode>
  <output>STRUCTURED</output>
  <anonStrCode>Y</anonStrCode>
  <cmapStrCode>Y</cmapStrCode>
  <fldArgs maxFields="4" allowed="ANY"/>
</function>

<function>
  <name>UCSFREQADDR</name>
  <attrTypes>MEMADDR</attrTypes>
  <equiStrCode>N</equiStrCode>
  <output>SIMPLE</output>
  <anonStrCode>Y</anonStrCode>
  <cmapStrCode>Y</cmapStrCode>
</function>
```

339
<stdFunction name="PASSTHRU" attrTypes="ANY" equiStrCode="Y"
output="SIMPLE" anonStrCode="Y" cmapStrCode="N">
  <fldArgs maxFields="1" allowed="ANY" />
</stdFunction>
</stdFunctions>

<cmpFunctions>
  - <cmpFunction name="DR1D1A">
    <input role="1" allowed="SIMPLE" />
  </cmpFunction>
  - <cmpFunction name="DR2D1A">
    <input role="1" allowed="SIMPLE" />
    <input role="2" allowed="SIMPLE" />
  </cmpFunction>
  - <cmpFunction name="DR3D1A">
    <input role="1" allowed="SIMPLE" />
    <input role="2" allowed="SIMPLE" />
    <input role="3" allowed="SIMPLE" />
  </cmpFunction>
  - <cmpFunction name="DR4D1A">
    <input role="1" allowed="SIMPLE" />
    <input role="2" allowed="SIMPLE" />
    <input role="3" allowed="SIMPLE" />
    <input role="4" allowed="SIMPLE" />
  </cmpFunction>
  - <cmpFunction name="DR1D1B">
    <input role="1" allowed="SIMPLE" />
  </cmpFunction>
  - <cmpFunction name="DR2D1B">
    <input role="1" allowed="SIMPLE" />
    <input role="2" allowed="SIMPLE" />
  </cmpFunction>
  - <cmpFunction name="DR3D1B">
    <input role="1" allowed="SIMPLE" />
    <input role="2" allowed="SIMPLE" />
    <input role="3" allowed="SIMPLE" />
  </cmpFunction>
  - <cmpFunction name="DR4D1B">
    <input role="1" allowed="SIMPLE" />
    <input role="2" allowed="SIMPLE" />
    <input role="3" allowed="SIMPLE" />
    <input role="4" allowed="SIMPLE" />
  </cmpFunction>
  - <cmpFunction name="DR1D1C">
    <input role="1" allowed="SIMPLE" />
  </cmpFunction>
  - <cmpFunction name="DR2D1C">
    <input role="1" allowed="SIMPLE" />
    <input role="2" allowed="SIMPLE" />
  </cmpFunction>
  - <cmpFunction name="DR3D1C">
    <input role="1" allowed="SIMPLE" />
    <input role="2" allowed="SIMPLE" />
    <input role="3" allowed="SIMPLE" />
  </cmpFunction>
  - <cmpFunction name="DR4D1C">
    <input role="1" allowed="SIMPLE" />
    <input role="2" allowed="SIMPLE" />
    <input role="3" allowed="SIMPLE" />
    <input role="4" allowed="SIMPLE" />
  </cmpFunction>
  - <cmpFunction name="DR1D2A">
    <input role="1" allowed="SIMPLE" />
  </cmpFunction>
  - <cmpFunction name="DR1D3A">
    <input role="1" allowed="SIMPLE" />
  </cmpFunction>
</cmpFunctions>
Appendix B. Algorithm validation rule syntax
<genFunction name="DTY4MM" equiStrCode="N" rbktStrCode="N" allowed="DATE" />
<genFunction name="DTMMDD" equiStrCode="N" rbktStrCode="N" allowed="DATE" />
</genFunctions>
</algorithm-rules>

Appendix B. Algorithm validation rule syntax 343
Appendix C. Understanding weight table names

The wgtcode values are automatically determined by the Master Data Engine using the following syntax:

\[
\text{mpi cmpspec cmpcode - mpi cmpspec cmpspeccode - mpi cmpfunc wgtsfx}<n> \\
\text{where } <n> \text{ is [1..8]}
\]

**Note:** Comparison functions may use 1 to 8 wgtsfx values as defined in mpi_cmpfunc.

A wgtcode in mpi wgthead links a single comparison function to a particular weight table. For example, say the HXW comparison function is added to the Guest algorithm and assigned the cmpspeccode value of 123. The Guest entity type, to which the Guest algorithm is associated, has a cmpcode of CMPGID (from the mpi_enttype table). In the mpi_cmpfunc table, the HXW comparison function has two non-blank wgtsfx values (wgtsfx1 = 2DIM and wgtsfx2 = SVAL), which means it requires two wgtcode entries in mpi_wgthead. Therefore, the Master Data Engine would require the following wgtcodes for this instance of the HXW comparison function in the Guest algorithm, based on the cmpcode - cmpspeccode - wgtsfx<\text{n}> formula:

CMPGID-123-2DIM

CMPGID-123-SVAL

Each wgtcode is used as a key to its associated weight values in one of the six pre-defined weight tables (see below). Additionally, each wgtcode must be unique across all comparison functions referenced by algorithms associated with a particular Master Data Engine instance. In order for the Master Data Engine to determine which weight table each wgtcode is associated with, the entries in mpi_wgthead also require a four-character wgttype value which maps to one of the weight tables:

<table>
<thead>
<tr>
<th>wgttype</th>
<th>Weight table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVAL</td>
<td>mpi_wgtsval</td>
<td>Used for string-value weights</td>
</tr>
<tr>
<td>NVAL</td>
<td>mpi_wgtnval</td>
<td>Used for integer-value weights</td>
</tr>
<tr>
<td>1DIM</td>
<td>mpi_wgt1dim</td>
<td>Used for 1-dimensional weights</td>
</tr>
<tr>
<td>2DIM</td>
<td>mpi_wgt2dim</td>
<td>Used for 2-dimensional weights</td>
</tr>
<tr>
<td>3DIM</td>
<td>mpi_wgt3dim</td>
<td>Used for 3-dimensional weights</td>
</tr>
<tr>
<td>4DIM</td>
<td>mpi_wgt4dim</td>
<td>Used for 4-dimensional weights</td>
</tr>
</tbody>
</table>

In the example above, the two weight tables used by the HXW comparison function are mpi_wgt2dim and mpi_wgtsval, which are indicated by its wgttypes of 2DIM and SVAL, respectively. However, the wgtsfx values for the other comparison functions do not always directly match one of the weight table types. For example, the BXNM comparison function has wgtsfx<\text{n}> values of XACT, META, INIT, and PSFX, which do not match any of the six standard wgttype values.
Appendix D. Globalization

IBM Initiate Workbench can be viewed in alternate languages. (For information on supported languages, contact your Initiate Systems project manager.) As is the case with all Initiate clients, if an organization has multi-language requirements, hub configuration can only be displayed in one language at a time. The language in which the application appears is dependent on the language specified in your Operating System (OS) settings; the default setting is U.S. English.

To accommodate translation into other languages, any dates entered in the application are in the ISO 8601 format (YYYY-MM-DD). The first four digits are always assumed by the software to be the year.

True dates returned from the Hub database are formatted based on the OS default. True dates are those dates required to be valid and complete. For example, the Last Modified Date in a U.S. English OS setting may display as 2004-08-14. An OS set to display French may display the date as 8 Jul 2004.

An assumption on the formatting of non-true dates—values that are not required to be valid or complete—cannot be made for a given locale. These dates are retrieved from the database and are displayed as they exist in the database.

All labels, such as buttons for Add and Edit, are retrieved from Hub Configuration resource bundles. Such labels display in the language setting of the Operating System. Labels such as Member Type are retrieved from the Hub database and are in the database-specified language. Entity and member types (e.g., Identity and Person) are labels defined in the hub database and cannot be displayed in the local language.

Some labels are combinations of storage locale and have the ability to be formatted appositionally.

Error messages sent to the OS (e.g. “No record found based on the input criteria”), and thus seen by users, display in the localized language. However, error messages sent to log files are currently only in U.S. English.
Appendix E. Creating cheat sheets for Initiate tools

About this task

As an Eclipse-based application, IBM Initiate Workbench supports the use of cheat sheets. Cheat sheets are custom dialogs that list the steps involved in performing specific tasks, and provide links that launch the IBM Initiate Workbench tools used to perform those steps. This appendix describes creating a basic cheat sheet and the various Initiate utilities you can execute through cheat sheets. For more detailed information about the built-in cheat sheet functionality, please refer to the Eclipse documentation, which you can find at http://www.ibm.com/developerworks/opensource/library/os-ecl-cheatsheets/

Creating a new cheat sheet

Procedure

1. From the main IBM Initiate Workbench menu, select File > New > Other....
2. On the New dialog, expand User Assistance and select Cheat Sheet.
3. Click Next.
4. On the New Cheat Sheet wizard, select the project into which you want to create the cheat sheet. If desired, you can place it into an existing folder within the project.
5. In the File name field, type a file name for storing the cheat sheet. Cheat sheets are saved in XML format, and the file name will automatically receive the .XML extension.
6. In the Cheat Sheet Type group box, select Simple Cheat Sheet.
7. Click Finish. A new file is created in the Navigator using the file name provided. The cheat sheet automatically opens in the Simple Cheat Sheet Editor.

Specifying commands for cheat sheet steps

About this task

Cheat sheets you create have enough built-in functionality to open in the cheat sheet dialog, but you must specify the commands each cheat sheet will execute. This is typically done using the Simple Cheat Sheet Editor, but advanced users can also edit the .XML file directly. The instructions below assume use of the Simple Cheat Sheet Editor.

Procedure

1. Edit an existing cheat sheet by right-clicking the cheat sheet’s xml file and selecting Open With > Simple Cheat Sheet Editor. The editor opens in the main editor pane.
2. In the Content pane, click the Title element. In the Definition pane, provide a title for your cheat sheet in the Title field. The text you type here is displayed at the very top of the cheat sheet.
3. In the Content pane, click the Introduction element. In the Definition pane, provide the introductory text about the cheat sheet in the Body field. If you want the introductory text to appear in bold text, retain the <b> and </b>
tags, and place your text between them. The text you type here is displayed below the Introduction subheading on the cheat sheet.

4. In the Content pane, click the **Item** element. In the Definition pane, provide the following information:
   - a title for the step in the Title field
   - a description of the step in the Body field. If you want the description to appear in bold text, retain the `<b>` and `</b>` tags, and place your text between them.

For each Item, the title and body text are displayed in order below the Introduction subheading.

5. In the Command pane, click the **Browse...** button. The Command Composer dialog opens.

6. Expand the Initiate entry and select the desired command. See “Initiate tools executable through cheat sheets” for a list of the commands and the functions they perform.

7. If the command takes a key/value map as an input variable (refer to the table above), type the variable name you plan to use in the format `${variable_name}` in the **Input variables** field (for example, `${valuesMap}`). Be sure to follow the steps under “Specifying an Input/Output Variable” on page 351 before attempting to run the cheat sheet.

8. Click **OK** to close the Command Composer dialog.

9. If additional steps are needed, click **Add Step**. A new **Item** element is created. Repeat steps 4-8 to assign the desired command to the step.

10. Save the project.

### Initiate tools executable through cheat sheets

The table below lists the commands and the functions you can execute through cheat sheets.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Initiate Project</td>
<td>Launches the New Initiate Project wizard. Returns a key/value map containing the project and Hub names.</td>
<td>none</td>
</tr>
<tr>
<td>Launch Algorithm Editor</td>
<td>Opens the Algorithm Editor.</td>
<td><strong>Input variables:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Key/value map containing the project and Hub names; value must be in the form <code>${map_name}</code></td>
</tr>
<tr>
<td>Launch Hub Configuration Editor</td>
<td>Opens the Hub Configuration Editor on the specified tab and sub-tab.</td>
<td><strong>Input variables:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Key/value map containing the project and Hub names; value must be in the form <code>${map_name}</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Primary tab default:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• a top-level tab to have focus in the editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Secondary tab default:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• a sub-tab to have focus in the editor</td>
</tr>
</tbody>
</table>
Table 71. Initiate tools executable through cheat sheets (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch an Initiate Wizard</td>
<td>Launches the desired Initiate wizard.</td>
<td>Input variables:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Key/value map containing the project and Hub names</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Initiate wizards:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Select one of the listed wizards to launch.</td>
</tr>
<tr>
<td>Run a Job</td>
<td>Launches the Job and Job Set wizard.</td>
<td>Input variables:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Key/value map containing the project and Hub names; value must be in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>form ${map_name}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default Job:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Select one of the listed jobs to include in a job set.</td>
</tr>
<tr>
<td>Select an Initiate Project and</td>
<td>Opens the Select project dialog. Returns a key/value</td>
<td>none</td>
</tr>
<tr>
<td>Hub</td>
<td>map containing the project and Hub names.</td>
<td></td>
</tr>
<tr>
<td>Show Analysis View</td>
<td>Opens the Analytics perspective and view.</td>
<td>Default Query:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Select one of the listed analysis queries to run.</td>
</tr>
</tbody>
</table>

Specifying an Input/Output Variable

About this task

The native behavior of the Simple Cheat Sheet Editor does not allow you to specify a variable to hold the return value for a command, which other commands may need as input. To remedy this, you need to add the output parameter manually using a text or XML editor.

Procedure

1. Right-click the cheat sheet’s xml file in the Navigator and select Open With > Text Editor. You may use your preferred text or XML editor instead, if desired.
2. Locate the line that identifies the Create a New Project or Select a Project command:
   
   ```xml
   <command serialization="NewProjectCommand" confirm="false">
   or
   ```
   ```xml
   <command serialization="ProjectSelectionCommand" confirm="false">
   3. Insert the parameter returns="variable name" where variable name is the variable you selected in step 7 under "Specifying commands for cheat sheet steps" on page 349. Be sure this matches exactly; the variable name is case-sensitive. In the example for step 7, we used the input variable ${valuesMap}. The text below shows how the valuesMap variable was added to the command line in the XML file.

   ```xml
   <xml version="1.0" encoding="UTF-8">
   <cheatsheet title="Select a project">
   <intro>
   <description>
   ```
<b>This cheat sheet opens the Select an Initiate Project dialog.</b>

This step opens the &quot;Select Initiate Project and Hub&quot; dialog, enabling...

4. Save the .XML file and close the text editor.

**Running an Initiate-supplied cheat sheet**

**Procedure**

1. Select Help > Cheat Sheets... from the main menu. The Cheat Sheet Selection dialog opens.
2. Choose the option to Select a cheat sheet from the list.
3. Open the Initiate folder and select the desired cheat sheet from the list. The cheat sheet opens.
4. Click the links in the order presented to execute the task's steps.

**Running your own cheat sheet**

**Procedure**

1. Select Help > Cheat Sheets... from the main menu. The Cheat Sheet Selection dialog opens.
2. Choose the option to Select a cheat sheet from a file.

**Note:** Cheat sheets that you create yourself do not appear in the list of available cheat sheets unless you package them into an Eclipse plugin, which is outside the scope of this document.

3. Click Browse... to locate the cheat sheet's .XML file. The cheat sheets are saved in the project, which is typically located in the path C:\Documents and Settings\username\workspace\project_name\.
4. Click Open on the Select Cheat Sheet Content File dialog.
5. Click OK on the Cheat Sheet Selection dialog. The cheat sheet opens.
6. Click the links in the order presented to execute the task's steps.
Appendix F. MPXDATA configuration file

The mpxdata configuration file (or “config file” as it is commonly called) is used by the Derive Data and Create UNLs (mpxdata) job (as well as mpxdata.exe) as a map for reading the customer’s extract file and a legend for converting the data to the Initiate database table layout. The configuration file tells mpxdata where each field is located in the extract and how to migrate it into the proper Initiate data format. The mpxdata utility can then perform member puts (inserts into the database) or member computes (parses out data into segment specific unload, or UNL files).

In order to build a configuration file, remember the following:

- Some data elements require more than one piece of information to constitute a complete Initiate record ready for insertion into the database. For example, Social Security Number requires both the actual ID and the ID Issuer. '111223333' would be the ID and 'SSA' would be the ID Issuer.
- Some data elements can be inserted using one or more data elements to make up a complete record. For example the Name field (LGLNAME above) can have up to seven elements that may be used to make up a single name record. If used, each one of these elements will have to be mapped from the source file to a database field inside an Initiate database record.
- Some fields, like date fields, can be represented in different ways (MMDDYYYY, YYYYMMDD, etcetera), and Derive Data and Create UNLs (mpxdata) must be told how the field is formatted inside the customer extract.

A configuration file is written for each unique data extract file. Therefore, the configuration files are typically designed by following the Data Extract Guide that the customer agreed to follow. Configuration files define several elements, including:

- Attributes to field number in the extract file
- String Cook options that remove leading & trailing data
- Methods that set the data type, like String or Date
- Field to field mapping for Attribute parts, like Last Name

Typically, the configuration file is named after the data extract file and is located in the IBM Initiate Workbench project directory.

The two examples below illustrate the mapping between a pipe-delimited data extract file containing actual data to be inserted into the Hub database, and the configuration file which defines how the data elements for each record are mapped.

CountyHospitalPatients.txt

CountyHospitalPatients.cfg

© Copyright IBM Corp. 1995, 2012
The following is a small example of a configuration file designed for delimited data:

<table>
<thead>
<tr>
<th>attrCode</th>
<th>ivar</th>
<th>offset</th>
<th>strcook</th>
<th>setData</th>
<th>constant</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEMHEAD</td>
<td>1</td>
<td>1</td>
<td>TR</td>
<td>SetString</td>
<td>srcCode</td>
<td></td>
</tr>
<tr>
<td>MEMHEAD</td>
<td>1</td>
<td>6</td>
<td>Z1</td>
<td>SetString</td>
<td>memIdnum</td>
<td></td>
</tr>
<tr>
<td>FACILITY</td>
<td>1</td>
<td>1</td>
<td>TR</td>
<td>SetString</td>
<td>attrVal</td>
<td></td>
</tr>
<tr>
<td>PTENTRID</td>
<td>1</td>
<td>1</td>
<td>TR</td>
<td>SetString</td>
<td>idIssuer</td>
<td></td>
</tr>
<tr>
<td>PTENTRID</td>
<td>1</td>
<td>5</td>
<td>ZR</td>
<td>SetString</td>
<td>idNumber</td>
<td></td>
</tr>
<tr>
<td>PTCHRTID</td>
<td>1</td>
<td>1</td>
<td>TR</td>
<td>SetString</td>
<td>idIssuer</td>
<td>## Patient Chart ID Issuer</td>
</tr>
<tr>
<td>PTCHRTID</td>
<td>1</td>
<td>6</td>
<td>ZR</td>
<td>SetString</td>
<td>idNumber</td>
<td>## Patient Chart ID Number</td>
</tr>
<tr>
<td>LGLNAME</td>
<td>1</td>
<td>7</td>
<td>TR</td>
<td>SetString</td>
<td>onmLast</td>
<td>## Patient Name</td>
</tr>
<tr>
<td>LGLNAME</td>
<td>1</td>
<td>8</td>
<td>TR</td>
<td>SetString</td>
<td>onmFirst</td>
<td>## Patient Name</td>
</tr>
<tr>
<td>LGLNAME</td>
<td>1</td>
<td>9</td>
<td>TR</td>
<td>SetString</td>
<td>onmMiddle</td>
<td>## Patient Name</td>
</tr>
<tr>
<td>LGLNAME</td>
<td>1</td>
<td>10</td>
<td>TR</td>
<td>SetString</td>
<td>onmSuffix</td>
<td>## Patient Name</td>
</tr>
<tr>
<td>BIRTHDT</td>
<td>1</td>
<td>11</td>
<td>NA</td>
<td>SetDate_Y4MD</td>
<td>dateVal</td>
<td>## Patient Birthdate</td>
</tr>
<tr>
<td>SEX</td>
<td>1</td>
<td>12</td>
<td>TR</td>
<td>SetString</td>
<td>attrVal</td>
<td>## Patient Sex</td>
</tr>
<tr>
<td>RACE</td>
<td>1</td>
<td>13</td>
<td>TR</td>
<td>SetString</td>
<td>attrVal</td>
<td>## Patient Race</td>
</tr>
<tr>
<td>SSN</td>
<td>1</td>
<td>0</td>
<td>NA</td>
<td>SetString</td>
<td>idIssuer</td>
<td>SSA</td>
</tr>
<tr>
<td>SSN</td>
<td>1</td>
<td>14</td>
<td>ZR</td>
<td>SetString</td>
<td>idNumber</td>
<td></td>
</tr>
<tr>
<td>HOMEADDR</td>
<td>1</td>
<td>15</td>
<td>TR</td>
<td>SetString</td>
<td>stLine1</td>
<td></td>
</tr>
</tbody>
</table>

_{Table 72. Sample mpxdata configuration file}_
Table 72. Sample mpxdata configuration file (continued)

<table>
<thead>
<tr>
<th>attrCode</th>
<th>ivar</th>
<th>offset</th>
<th>strcook</th>
<th>setData</th>
<th>constant</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOMEADDR</td>
<td>1</td>
<td>16</td>
<td>TR</td>
<td>SetString</td>
<td>city</td>
<td>## Patient Home Address</td>
</tr>
<tr>
<td>HOMEADDR</td>
<td>1</td>
<td>17</td>
<td>TR</td>
<td>SetString</td>
<td>state</td>
<td>## Patient Home Address</td>
</tr>
<tr>
<td>HOMEADDR</td>
<td>1</td>
<td>18</td>
<td>TR</td>
<td>SetString</td>
<td>zipcode</td>
<td>## Patient Home Address</td>
</tr>
<tr>
<td>HOMEPHON</td>
<td>1</td>
<td>19</td>
<td>TR</td>
<td>SetString</td>
<td>phNum</td>
<td>ber</td>
</tr>
<tr>
<td>LANGUAGE</td>
<td>1</td>
<td>20</td>
<td>TR</td>
<td>SetString</td>
<td>attrVal</td>
<td>## Language</td>
</tr>
<tr>
<td>RELIGION</td>
<td>1</td>
<td>21</td>
<td>TR</td>
<td>SetString</td>
<td>attrVal</td>
<td>## Religion</td>
</tr>
<tr>
<td>GTNAME</td>
<td>1</td>
<td>22</td>
<td>NA</td>
<td>SetString</td>
<td>onm</td>
<td>Last</td>
</tr>
<tr>
<td>GTNAME</td>
<td>1</td>
<td>23</td>
<td>NA</td>
<td>SetString</td>
<td>onm</td>
<td>First</td>
</tr>
<tr>
<td>GTDOB</td>
<td>1</td>
<td>24</td>
<td>NA</td>
<td>SetDate_Y4MD</td>
<td>dateVal</td>
<td>## Guarantor Birthdate</td>
</tr>
<tr>
<td>GTSSN</td>
<td>1</td>
<td>0</td>
<td>NA</td>
<td>SetString</td>
<td>idIssuer</td>
<td>SSA</td>
</tr>
<tr>
<td>GTSSN</td>
<td>1</td>
<td>25</td>
<td>ZR</td>
<td>SetString</td>
<td>idNum</td>
<td>ber</td>
</tr>
<tr>
<td>GTHADDR</td>
<td>1</td>
<td>26</td>
<td>NA</td>
<td>SetString</td>
<td>stLine1</td>
<td>## Guarantor Home Address</td>
</tr>
<tr>
<td>GTHADDR</td>
<td>1</td>
<td>27</td>
<td>NA</td>
<td>SetString</td>
<td>city</td>
<td>## Guarantor Home Address</td>
</tr>
<tr>
<td>GTHADDR</td>
<td>1</td>
<td>28</td>
<td>NA</td>
<td>SetString</td>
<td>state</td>
<td>## Guarantor Home Address</td>
</tr>
<tr>
<td>GTHADDR</td>
<td>1</td>
<td>29</td>
<td>NA</td>
<td>SetString</td>
<td>zipcode</td>
<td>## Guarantor Home Address</td>
</tr>
<tr>
<td>GTINSNM</td>
<td>1</td>
<td>30</td>
<td>TR</td>
<td>SetString</td>
<td>attrVal</td>
<td>## Guarantor Ins Name</td>
</tr>
<tr>
<td>GTINSPOL</td>
<td>1</td>
<td>31</td>
<td>TR</td>
<td>SetString</td>
<td>attrVal</td>
<td>## Guarantor Ins Policy Number</td>
</tr>
<tr>
<td>GTINSCOV</td>
<td>1</td>
<td>32</td>
<td>TR</td>
<td>SetString</td>
<td>attrVal</td>
<td>## Guarantor Ins Coverage</td>
</tr>
<tr>
<td>GTEMPNM</td>
<td>1</td>
<td>33</td>
<td>TR</td>
<td>SetString</td>
<td>attrVal</td>
<td>## Guarantor Employer Name</td>
</tr>
<tr>
<td>GTOCC</td>
<td>1</td>
<td>34</td>
<td>TR</td>
<td>SetString</td>
<td>attrVal</td>
<td>## Guarantor Occupation</td>
</tr>
</tbody>
</table>

**attrCode**
This column is used to identify which data element inside the database you are going to populate. This name must match the attribute’s “ATTRCODE” inside the mpi_segattr table.

**ivar (instance variable number)**
This column defines how many of the same named data elements are in this customer record. For example, if the extract contains three phone numbers and inside mpi_segattr you only have one attribute defined as PHONE (rather than WRKPH, HOMEPH, CELLPH), you would increment the ivar column for each
PHONE entry you have in the configuration file. Create three separate attributes inside mpi_segattr to handle the three phone types mentioned above.

offset (Position)
This is the position of the field within the customer extract to insert into the Initiate database record. An offset of 0 indicates we will be inserting a constant value in this field, and not pulling the value from the extract. The example above is inserting a constant value of SSA as the ID Issuer of the SSN.

length
If the Input File Format for the job is set to Fixed, the length in bytes of the field must be specified in the fourth column position. If the Input File Format is set to Delimited, the length column should be omitted from the configuration file. Below is an example of a configuration file whose Input File Format is fixed:

<table>
<thead>
<tr>
<th>attrCode</th>
<th>ivar</th>
<th>offset</th>
<th>Length</th>
<th>strcook</th>
<th>setData</th>
<th>constant</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM HEAD</td>
<td>1</td>
<td>0</td>
<td>0 NA</td>
<td>SetString</td>
<td>srcCode</td>
<td>RMC</td>
<td></td>
</tr>
<tr>
<td>MEM HEAD</td>
<td>1</td>
<td>6</td>
<td>Z1</td>
<td>SetString</td>
<td>memId</td>
<td>num</td>
<td></td>
</tr>
<tr>
<td>LGL NAME</td>
<td>1</td>
<td>7</td>
<td>25 TR</td>
<td>SetString</td>
<td>onmLast</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LGL NAME</td>
<td>1</td>
<td>32</td>
<td>25 TR</td>
<td>SetString</td>
<td>onmFirst</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOME ADDR</td>
<td>1</td>
<td>57</td>
<td>20 ZR</td>
<td>SetString</td>
<td>stLine1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOME ADDR</td>
<td>1</td>
<td>77</td>
<td>20 TR</td>
<td>SetString</td>
<td>city</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOME ADDR</td>
<td>1</td>
<td>97</td>
<td>2 TR</td>
<td>SetString</td>
<td>state</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOME ADDR</td>
<td>1</td>
<td>99</td>
<td>5 TR</td>
<td>SetString</td>
<td>zipCode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOME ADDR</td>
<td>1</td>
<td>104</td>
<td>3 TR</td>
<td>SetString</td>
<td>phArea</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOME ADDR</td>
<td>1</td>
<td>107</td>
<td>8 TR</td>
<td>SetString</td>
<td>phNumber</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

strcook (Transform)
This is the optional edit method or “cook” method you can apply to the customer data element before inserting it into the Initiate database. These are very basic, and can be used to remove blanks or zeros from the right or left side of the data element if necessary. The cook and setData methods can be displayed on the command line using the -methods option.

- NA -- do nothing
- TR -- trim leading and trailing blanks, allow empty string
- BL -- trim blanks from the left, allow empty string
- ZL -- trim zeroes from the left, allow empty string
B1 -- trim blanks from the left, leave at most 1
Z1 -- trim zeroes from the left, leave at most 1
BR -- trim blanks from the right, allow empty string
ZR -- trim zeroes from the right, allow empty string
ZX -- trim only if all zeroes, creating empty string

**setData**
This is the database method used to populate the Initiate record with the customer data. Each data type has its own set of SETDATA methods. You must take care to select the proper one for the data element you are inserting. The cook and setData methods can be displayed on the command line using the -methods option.

The available SETDATA methods are:
- SetString
- SetNumber
- SetDate_MDY4
- SetDate_MDY2
- SetDate_MDCY2
- SetDate_Y4MD
- SetDate_Y2MD
- SetDate_PMDY
- SetDate_PYMD
- SetDate_DMY4

**Note:** It should be noted that the SetDate* format methods work on 'date' input data, but the destination in the database is expected to be a String (character field).

The setData method is followed by the column name into which you wish to insert the data.

**constant (optional)**
If present, this represents a constant value that will be inserted into the Initiate database for the column indicated with setData. It does not physically appear in the record, but behaves as though it did. If used, the offset (and length, if applicable) should be set to 0. For example:

```
SSN  1  0  NA  SetString  idIssuer  SSA  ##  Patient SSN
```

The string “SSA” will be written to the idIssuer column of the SSN attribute. You can also use the constant to set the asaIdxNo field. For example:

<table>
<thead>
<tr>
<th>attrCode</th>
<th>ivar</th>
<th>offset</th>
<th>strcook</th>
<th>setData</th>
<th>constant</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>1</td>
<td>0</td>
<td>NA</td>
<td>SetString</td>
<td>asaldxNo</td>
<td>1</td>
</tr>
<tr>
<td>NAME</td>
<td>1</td>
<td>5</td>
<td>TR</td>
<td>SetString</td>
<td>onmFirst</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>1</td>
<td>6</td>
<td>TR</td>
<td>SetString</td>
<td>onmMiddle</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>1</td>
<td>7</td>
<td>TR</td>
<td>SetString</td>
<td>onmLast</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>2</td>
<td>0</td>
<td>NA</td>
<td>SetString</td>
<td>asaldxNo</td>
<td>2</td>
</tr>
<tr>
<td>NAME</td>
<td>2</td>
<td>8</td>
<td>TR</td>
<td>SetString</td>
<td>onmFirst</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>2</td>
<td>9</td>
<td>TR</td>
<td>SetString</td>
<td>onmMiddle</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>2</td>
<td>10</td>
<td>TR</td>
<td>SetString</td>
<td>onmLast</td>
<td></td>
</tr>
</tbody>
</table>
Any comment you wish to make about a field should be placed at the end of the row preceded by a hash (pound) symbol. This comment is not saved in the Initiate database.

<table>
<thead>
<tr>
<th># Patient Information</th>
<th></th>
<th></th>
<th>SetString idNumber</th>
<th></th>
<th># Patient SSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>1</td>
<td>0</td>
<td>NA</td>
<td></td>
<td># Patient SSN</td>
</tr>
</tbody>
</table>
Legal Statement

Licensed Materials – Property of IBM

- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with
IBM Corp. IBM, the IBM logo, Initiate, and Initiate Master Data Service are
trademarks of IBM Corp., registered in many jurisdictions worldwide. Java and all
Java-based trademarks and logos are trademarks or registered trademarks of
Oracle and/or its affiliates. Other product and service names might be trademarks
of IBM, or other companies. This Program is licensed under the terms of the
license agreement accompanying the Program. This license agreement may be
either located in a Program directory folder or library identified as "License" or
"Non-IBM License", if applicable, or provided as a printed license agreement.
Please read this agreement carefully before using the Program. By using the
Program, you agree to these terms.
**Notices and trademarks**

This information was developed for products and services offered in the U.S.A.

**Notices**

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web
sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to
IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

**Trademarks**

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

Other product and service names might be trademarks of IBM or other companies.


The following terms are trademarks or registered trademarks of other companies:

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
### Algorithms

**Algorithm Comparison**
- Active 105
- Adding 105
- Anonymous values 251
- Assigning to member type 106
- Configuring 2
- Copying between projects 106

**Algorithm Editor**
- About 8
- Adding tooltips and labels 167
- Algorithm tab 157
- Attributes 158
- Bucketing functions
  - Properties 162
  - Setting 163
- Bucketing functions, adding and deleting 162
- Bucketing groups
  - Adding or deleting 165
  - Properties 166
  - Setting 166
- Comparison functions
  - Properties 164
  - Setting 165
- Comparison functions, adding and deleting 161
- Components 166
- Palette 156
- Properties 157
- Query roles, adding and deleting 162
- Standardization functions
  - Properties 159
  - Setting 159
  - Two standardization roles 160
- Standardization functions, adding and deleting 158
- Sticky selection 153
- Swimlane width 153
- Thresholds and weight properties
  - Tab 157
  - Using 153
- Weight generation parameters 167
- Algorithm tab 157

**Algorithms (continued)**
- Copying (continued)
  - Within projects 105
- Deleting 106
- Display top level properties 157
- Editing 105, 155
- Entity type/member type associations 154
- Equi values 251
- Generation functions 328
- Loading 154
- Renaming 155
- Sample validation rules XML 337
- Saving 154
- See also bucketing functions 327
- See also comparison functions 289
- See also standardization functions 250
- Validation 335

### Analytics

- About 187
- Adding queries to views 189
- Attribute Completeness query 194
- Attribute Validity query 194
- Bucket analysis 194
- Bucket Analysis Overview query 194
- Bucket Composition query 195
- Bucket Size Distribution query 195
- Buckets By Size query 195
- Bulk Cross Match Comparison Distribution query 196
- Comparison Data Completeness query 194
- Connecting to a hub 188
- Data analysis queries 194
- Entities By Size query 191
- Entity analysis 191
- Entity Composition query 191
- Entity Size Distribution query 192
- Exporting query results 189
- Linkage analysis queries 197
- Loading previous results 190
- Member Bucket Frequency query 196
- Member Comparison Distribution query 196
- Member Comparisons query 192
- Member Duplicates query 197
- Member Entity Frequency query 193
- Member Entity Values query 193
- Member Overlaps query 197
- Members By Bucket Count query 197
- Members by Entity Count query 193
- Obtaining data 190
- Preparing data for large datasets 191
- Score Distribution query 193
- View 7, 187, 188

### Anonymous Value Utility
- About 142
- Launching 143
- Using 143
- Rerunning 201
- Application properties
  - Adding 128
  - Deleting 129
  - Editing 129
  - Properties 128
- Applications
  - Adding 127
  - Configuring in Workbench 126
  - Deleting 127
  - Editing 127
  - Properties 127
- Attribute Completeness query 194
- Attributes
  - About 91
  - Adding 92
  - Deleting 89, 93
  - Editing 93
  - Field properties 108
  - Properties 91
- Auditing view 151
- Autolink
  - Setting thresholds 173
- Alternate Languages 347

---

© Copyright IBM Corp. 1995, 2012
bucket generation functions (continued)
  SORTED 334
  SRANGE 333
Bucket Size Distribution query 195
bucketing functions
  adding and deleting 162
  ADDR2 328
  ATTR 328
  bucketing 327
  CDATE 328
  CXNM 328
  DATE 328
  properties 162
  PXNM 328
  setting 163
  validation rules 336
bucketing groups
  adding or deleting 165
  properties 166
  setting 166
Buckets By Size query 195
bulk cross match
  about 23
Bulk Cross Match Comparison
  Distribution query 196
Bulk Tools jobs
  Compare Members in Bulk (mpxcomp) 40
  Derive Data and Create UNLs (mpxdata) 49
  Derive Data from Hub (mpxredvd) 56
  Derive Data from UNLs (mpxfsdvd) 59
  Export Linkage History (mpxxeia) 63
  Link Entities (mpxlink) 64
  Link Relationships 71
  Prepare Binary Files (mpxprep) 73
  Sort Binary Link Data (mpxsort) 76

C
callout handlers
  about 199
  adding external libraries 200
  adding registrations 204
  Clover.ETL considerations 203
  creating 199
  creating launch configuration 201
  custom 3
  deleting registrations 205
  deploying 205
  editing registrations 205
  enabling 199
  packaging the handler 200
  registering .NET handlers 202
  registering java handlers 202
  registering the handler 201
  testing 205
cheat sheets
  about 4
  commands and handlers 349
  create 349
  creating 349
  executables 350
  output variables 351
cheat sheets (continued)
  running
    built-in 352
    custom 352
  clerical review
    setting thresholds 173
  Clover.ETL
    callout handler packaging 203
  composite algorithm functions
    enabling and disabling 153
    mapping to classic functions 247
  composite views
    about 97
    adding 100
    deleting 101
    editing 101
    permissions 146
    properties 98
    setting group access to 101
  configuration comparison
  configuration editor
    assigning users to groups 151
    audit logging 152
    interactions 147
    operations
      guidelines 149
      permissions 149
      relationship attributes 150
  configuration file
    MPXDATA 353
  configurations
    comparing 7
  Configure Update Synchronization
Service
  Flexible Search 83
  configuring
    Inspector 231
    custom phonetics 281
    custom task summary
      Inspector configuration 233
  customer support
    contacting 367

D
data analysis queries 194
data derivation 21
database tools
  jobs 78
Database Tools jobs
  Load UNLs to DB (madunlload) 78
  Unload UNLS from DB (madunlunload) 80
default search configuration
  creating 241
definitional sources
  adding 103
deleting configuration objects 89
deleting users
  user management 210
Deploy Hub Configuration 83
deploying
  configurations 3
deploying projects 17
Derive Data and Create UNLs (mpxdata) 49
Derive Data from Hub (mpxredvd) 56
Derive Data from UNLs (mpxfsdvd) 59
deriving data 21

e
edit distance comparison 290
Inspector configuration (continued)
label pattern 236
member types 233
search forms
adding 237
removing 237
search result forms
adding 238
setting minimum score 239
Inspector configuration editor
about 8
interaction codes 203
interactions
permissions 147
Internal LDAP server 207

J
JMX browser, accessing 221
JMX browser, administrative actions 226
JMX browser, log4j node 223
JMX browser, threadpools node 225
JMX browser, listeners node 222
job categories 25
job sets
adding jobs to existing 87
creating 23
introduction 21
rerunning 87
running 24
saving 24
templates 87
jobs
about 3
adding jobs to existing job set 87
algorithm & analysis 25
bulk tools 39
Compare Members (mpmcomp) 26
Compare Members in Bulk (mpxcomp) 40
Configure Update Synchronization Service 83
custom work directory 25
database tools 78
Deploy Hub Configuration 83
Derive Data and Create UNLs (mpxdata) 49
Derive Data from Hub (mpxredvd) 56
Derive Data from UNLs (mpxfsvdvd) 59
Execute Remote Graph 84
Export Linkage History (mpxseia) 63
files 25
full text index 82
Generate Frequency Stats (mpxfreq) 27
Generate Threshold Analysis Pairs 30
generate weights 33
get file 87
hub administration 83
Index by Members 82
Index by UNLs 83
introduction 21
Link Entities (mpxlink) 64
Link Relationships 71
jobs (continued)
Load UNLs to DB (madunlload) 78
local graph
exporting 86
mpxrule 39
Prepare Binary Files (mpxprep) 73
prepare binary bucket analysis 37
Resume Hub 87
Sort Binary Link Data (mpxsord) 76
Suspend Hub 87
Unload UNLs from DB (madunlload) 80
validate weights 38
validating weights 179
view 7
Member Comparison Distribution query 196
Member Comparisons query 192
Member Duplicates query 197
Member Entity Frequency query 193
Member Entity Values query 193
Member Overlaps query 197

L
language support 347
launch configuration 201
LDAP
about 207
adding new users 214
adding users to groups 216
Bind DN 212
group management 213
importing/exporting 213
internal LDAP server 207
multi-byte characters
password 215
user name 215
perspective 211
pre-configured groups 207, 208
system user 208
user management 214
legal notices 361
level 1 validation rules 180
level 2 validation rules 183
libraries, external 200
Link Entities (mpxlink) 64
Link Relationships 71
linkage analysis queries 197
linkage statuses
adding 119
deleting 120
editing 120
properties 119
linkage types 119
properties 117
linkages 116
listeners node 222
Load UNLs to DB (madunlload) 78
loading previous analysis results 190
local graph
exporting 86
local weight validation 179
log in to the Hub 4
log4j node 223
Obtaining analytics data 190
operations
guidelines 149
setting permissions 149

O
obtaining analytics data 190
operations
permissions
setting for groups 146
perspectives
about 5
phonetic name comparison 290
prefix and compound comparison 291
Prepare Binary Files (mpxprep) 73

P
pair manager
filtering by status 228
sample pair file 227
selecting data columns 229
using 228
permissions
setting for groups 146
perspectives
about 5
phonetic name comparison 290
prefix and compound comparison 291

M
mbean explorer
view 7, 221
Member Bucket Frequency query 196
member bucket values
analytics 196

N
navigating the LDAP perspective 211
navigating the user management perspective 209
navigator view
about 6
nickname comparison 290
nickname-meta comparison 290
T

- task status
  - adding 116
  - editing 116
- task statuses
  - deleting 116
  - properties 114
- task types
  - adding 115
  - editing 115
  - properties 113
- tasks 113
  - deleting 115
  - validation 3
- testing callout handlers 205
- threadpools node 225
- threshold calculator using 173
- thresholds
  - auto-link 173
  - calculating 173
  - clerical review 173
  - setting manually 173
- thresholds and weight properties tab 157
- trademarks
  - list of 361
- translation 347

U

- undo
  - about 4
- Unload UNLS from DB (madununload) 80
- update manager
  - about 1
  - upgrading projects 18
- user management
  - about 207
  - adding groups 213
  - assigning users to groups 210
  - deleting users 210
  - editing users 210
  - group DN containers 213
  - perspective 209
  - removing users from groups 211
- users
  - adding 209

V

- validate weights 38
- validating weights
  - about 179
  - job 179
  - locally 179
  - validation 3
- validation rules
  - adding 335
  - bucketing functions 336
  - comparison functions 336
  - generation functions 336
  - level 1 180
  - sample XML 337
  - standardization functions 335

- validation tasks
  - problems view 3
- versioning
  - about 4
- views
  - about 5
  - analytics 187
  - auditing 151
  - console 6
  - groups 145
  - jobs 7
  - mbean explorer 7, 221
  - navigator 6
  - problems 6
  - validation tasks 3
  - properties 6
  - search 7

W

- web reports
  - adding 130
  - properties 129
- weight generation
  - about 176
  - frequency-based bucketing 176
  - parameters 167
  - weight table names 345
- weight tables
  - wgttypes 345
- weight validation
  - about 179
  - AXP 1dim rule 185
  - AXP sval rule 183
  - basic 1dim rule 181
  - basic 2dim rule 181
  - basic 3dim rule 182
  - basic nval rule 180
  - basic sval rule 180
  - CXNM sval rule 183
  - DOB 1dim rule 185
  - DOB nval rule 183
  - gender sval rule 184
  - level 2 rules 183
  - local 179
  - PXNM sval rule 184
  - QXNM sval rule 184
  - rules 180
  - SSN 1dim rule 186
  - ZAP 3dim rule 186
- weights
  - generating
    - introduction 175
    - steps 177
  - generation properties 171
  - loading frequency data 142
  - threshold calculator 173
  - validating 179

- word mismatch 291
- Workbench
  - adding users 209
  - basics 1
  - concepts 2
  - screen layout 4
- Workbench projects managing 13
Contacting IBM

You can contact IBM for customer support, software services, product information, and general information. You also can provide feedback to IBM about products and documentation.

The following table lists resources for customer support, software services, training, and product and solutions information.

Table 73. IBM resources

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description and location</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Support Portal</td>
<td>You can customize support information by choosing the products and the topics that interest you at <a href="http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/IBMInitiate_Master_Data_Service">www.ibm.com/support/entry/portal/Overview/Software/Information_Management/IBMInitiate_Master_Data_Service</a></td>
</tr>
<tr>
<td>Software services</td>
<td>You can find information about software, IT, and business consulting services, on the solutions site at <a href="http://www.ibm.com/businesssolutions/">www.ibm.com/businesssolutions/</a></td>
</tr>
<tr>
<td>My IBM</td>
<td>You can manage links to IBM web sites and information that meet your specific technical support needs by creating an account on the My IBM site at <a href="http://www.ibm.com/account/">www.ibm.com/account/</a></td>
</tr>
<tr>
<td>Training and certification</td>
<td>You can learn about technical training and education services designed for individuals, companies, and public organizations to acquire, maintain, and optimize their IT skills at <a href="http://www.ibm.com/software/sw-training/">http://www.ibm.com/software/sw-training/</a></td>
</tr>
</tbody>
</table>

Providing feedback

The following table describes how to provide feedback to IBM about products and product documentation.

Table 74. Providing feedback to IBM

<table>
<thead>
<tr>
<th>Type of feedback</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product feedback</td>
<td>You can provide general product feedback through the Consumability Survey at <a href="http://www.ibm.com/software/data/info/consumability-survey">www.ibm.com/software/data/info/consumability-survey</a></td>
</tr>
<tr>
<td>Type of feedback</td>
<td>Action</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Documentation feedback</td>
<td>To comment on the information center, click the Feedback link on the top right side of any topic in the information center. You can also send comments about PDF file books, the information center, or any other documentation in the following ways:</td>
</tr>
<tr>
<td></td>
<td>• Online reader comment form: <a href="http://www.ibm.com/software/data/rcf/">www.ibm.com/software/data/rcf/</a></td>
</tr>
<tr>
<td></td>
<td>• E-mail: <a href="mailto:comments@us.ibm.com">comments@us.ibm.com</a></td>
</tr>
</tbody>
</table>