IBM Security Identity Manager
Version 6.0

Reference Guide

IBM
Note

Before using this information and the product it supports, read the information in "Notices" on page 269.

Edition notice

Note: This edition applies to version 6.0 of IBM Security Identity Manager (product number 5724-C34) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 2012, 2013.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Table of contents

Table list ................................ vii

About this publication ........................ ix
Access to publications and terminology .. ix
Accessibility ................................... x
Technical training ............................ x
Support information .......................... x
Statement of Good Security Practices .......... x

Chapter 1. Accessibility features for IBM Security Identity Manager ........ 1

Chapter 2. Application extensions . . . 3
WorkflowApplication interface ........ 3
Application extension methods ........ 3
Registering extensions ........... 4

Chapter 3. Application programming interfaces .......... 7
Applications API ......................... 7
Self registration API ....................... 8
Access control information list (ACI) API ...... 8
IBM Security Identity Manager group API .... 8
Provisioning policy API ................... 9
Recertification policy API .................. 9
Reconciliation API ......................... 9
Authentication API ........................ 10
Data services API .......................... 10
IBM Directory Integration API ............ 10
JavaScript API ............................. 10
Mail API .................................. 11
Password rules API ........................ 11
Policy analysis API ........................ 11
Service provider API ...................... 11
Shared access APIs ....................... 12
Shared access application APIs ........ 12
Shared access web services API .......... 14
Shared access authorization extension APIs . 14
Shared access JavaScript APIs ........ 14
Single sign-on API ....................... 14
Web services API .......................... 15
Workflow API ............................. 18

Chapter 4. Dictionary for a password policy ................ 19

Chapter 5. Dataservices attributes for recertification ........ 21

Chapter 6. Workflow extensions for recertification ........ 23

Chapter 7. Date range customization ........ 27

Chapter 8. Dynamic tags in mail templates ........ 29
Mail templates ........................... 34
Manual service default messages .......... 35
Recertification default messages .......... 36
Workflow default messages .......... 42

Chapter 9. JavaScript extensions overview ........ 53
Packaged extensions ....................... 54
AttributesExtension ....................... 54
DelegateExtension ......................... 55
EmailContextExtension ................... 55
EnroleExtension ........................... 55
IdentityPolicyExtension .................. 55
LoopCountExtension ....................... 56
Model extensions package ................ 56
PersonPlacementRulesExtension ........... 58
PostOfficeExtension ....................... 58
ProvisioningPolicyExtension ............. 58
ReminderExtension ....................... 58
ServiceExtension ......................... 59
SubjectExtension ......................... 59
WorkflowExtension ....................... 59
Registering JavaScript extensions .......... 60
Configuring scriptframework.properties .... 62
 Migration of custom FESI extensions to the IBM JSEngine ......................... 63
Best practice in handling function returns 63
Plain Old Java Object (POJO) example ........ 63
Conversion to a script extension .......... 65
Creation of a constructor .................. 66
Download of fesi.jar from a non-IBM source (deprecated) .......... 67

Chapter 10. JavaScript extension reference ................ 69
How to read the reference pages ........... 70
Account .................................. 72
AccountSearch ............................ 72
AccountSearch.searchByOwner() .......... 72
AccountSearch.searchByUid() .......... 73
| 1. | Syntax and example of using JavaScript code to replace message content | 29 |
| 2. | Syntax and examples of using a RE tag to replace message content | 30 |
| 3. | Syntax and example of using tags to replace message content | 31 |
| 4. | Syntax and examples of ITIMURL | 31 |
| 5. | Escape characters | 32 |
| 6. | Host components and script extensions | 53 |
| 7. | Script class keys | 62 |
| 8. | Script extensions | 69 |
| 9. | Provisioning policy examples | 159 |
| 10. | Sample provisioning policies | 159 |
| 11. | SubForm parameters | 167 |
| 12. | SubForm parameters | 168 |
| 13. | Properties files | 171 |
| 14. | Non-modifiable properties files | 173 |
| 15. | adhocreporting.properties properties | 175 |
| 16. | DataBaseFunctions.conf | 183 |
| 17. | enRoleAuditing.properties properties | 184 |
| 18. | enRoleAuthentication.properties properties | 186 |
| 19. | enRoleDatabase.properties properties | 188 |
| 20. | enRoleLDAPConnection.properties properties | 191 |
| 21. | enRoleLogging.properties properties | 194 |
| 22. | enRoleMail.properties properties | 205 |
| 23. | enrolepolicies.properties properties | 208 |
| 24. | enroleworkflow.properties properties | 211 |
| 25. | fesiextensions.properties properties (deprecated) | 213 |
| 26. | helpmappings.properties properties | 214 |
| 27. | reporttabledeny.properties | 214 |
| 28. | SelfServiceHelp.properties | 217 |
| 29. | SelfServiceUL.properties | 218 |
| 30. | ui.properties properties | 221 |
| 31. | WebSphere application server properties | 229 |
| 32. | Remote services properties | 233 |
| 33. | Web services properties | 233 |
| 34. | Application server properties | 235 |
| 35. | Organization properties | 236 |
| 36. | LDAP server properties | 237 |
| 37. | Search and LDAP control properties | 238 |
| 38. | Person profile property | 240 |
| 39. | Profile and schema cache properties | 241 |
| 40. | Messaging properties | 242 |
| 41. | Scheduling properties | 244 |
| 42. | Password transaction monitor properties | 244 |
| 43. | XML and DTD properties | 245 |
| 44. | LDAP connection pool properties | 245 |
| 45. | Encryption properties | 247 |
| 46. | Challenge response encoding properties | 249 |
| 47. | System configuration properties | 249 |
| 48. | Mail services properties | 250 |
| 49. | Workflow configuration properties | 251 |
| 50. | Reconciliation properties | 257 |
| 51. | Shared secret hashing properties | 260 |
| 52. | Lifecycle rule properties | 261 |
| 53. | Product property | 261 |
| 54. | Application client request properties | 262 |
| 55. | Reverse password synchronization properties | 262 |
| 56. | Post office properties | 262 |
| 57. | Database resource bundle properties | 264 |
| 58. | Database cleanup properties | 264 |
| 59. | Account restore properties | 264 |
| 60. | Create password check box default properties | 265 |
| 61. | Access catalog properties | 265 |
| 62. | Default identity feed properties | 266 |
| 63. | Default upgrade properties | 267 |
| 64. | Multiple password-synch agent properties | 267 |
About this publication

IBM Security Identity Manager Reference Guide provides reference information that is organized to help locate particular facts quickly, such as the description of a certain command-line parameter. For your convenience, reference information is gathered and categorized for easy access.

Access to publications and terminology

This section provides:
- A list of publications in the IBM Security Identity Manager library.
- Links to "Online publications."
- A link to the "IBM Terminology website" on page x.

IBM Security Identity Manager library

The following documents are available in the IBM Security Identity Manager library:
- IBM Security Identity Manager Quick Start Guide, CF3L2ML
- IBM Security Identity Manager Planning Guide, GC14-7694-01
- IBM Security Identity Manager Configuration Guide, SC14-7696-01
- IBM Security Identity Manager Administration Guide, SC14-7701-01
- IBM Security Identity Manager Troubleshooting Guide, GC14-7702-01
- IBM Security Identity Manager Error Message Reference, GC14-7393-01
- IBM Security Identity Manager Database and Directory Server Schema Reference, SC14-7395-01
- IBM Security Identity Manager Glossary, SC14-7397-01

Online publications

IBM posts product publications when the product is released and when the publications are updated at the following locations:

IBM Security Identity Manager library

The product documentation site displays the welcome page and navigation for the library.


IBM Security Systems Documentation Central

IBM Security Systems Documentation Central provides an alphabetical list of all IBM Security Systems product libraries and links to the online documentation for specific versions of each product.
IBM Publications Center

The IBM Publications Center site offers customized search functions to help you find all the IBM publications you need.

IBM Terminology website


Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully. With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate all features of the graphical user interface.

For additional information, see the topic "Accessibility features for IBM Security Identity Manager" in the IBM Security Identity Manager Reference Guide.

Technical training

For technical training information, see the following IBM Education website at http://www.ibm.com/software/tivoli/education.

Support information

If you have a problem with your IBM® software, you want to resolve it quickly. IBM provides the following ways for you to obtain the support you need:

Online

Go to the IBM Software Support site at http://www.ibm.com/software/support/probsub.html and follow the instructions.

IBM Support Assistant

The IBM Support Assistant (ISA) is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. The ISA provides quick access to support-related information and serviceability tools for problem determination. To install the ISA software, see the IBM Security Identity Manager Installation Guide. Also see: http://www.ibm.com/software/support/isa

Troubleshooting Guide

For more information about resolving problems, see the IBM Security Identity Manager Troubleshooting Guide.

Statement of Good Security Practices

IT system security involves protecting systems and information through prevention, detection and response to improper access from within and outside your enterprise. Improper access can result in information being altered, destroyed, misappropriated or misused or can result in damage to or misuse of your systems, including for use in attacks on others. No IT system or product should be considered completely secure and no single product, service or security measure can be completely effective in preventing improper use or access. IBM systems,
products and services are designed to be part of a comprehensive security approach, which will necessarily involve additional operational procedures, and may require other systems, products or services to be most effective. IBM DOES NOT WARRANT THAT ANY SYSTEMS, PRODUCTS OR SERVICES ARE IMMUNE FROM, OR WILL MAKE YOUR ENTERPRISE IMMUNE FROM, THE MALICIOUS OR ILLEGAL CONDUCT OF ANY PARTY.
Chapter 1. Accessibility features for IBM Security Identity Manager

Accessibility features help users who have a disability, such as restricted mobility or limited vision, to use information technology products successfully.

**Accessibility features**

IBM Security Identity Manager conforms to Section 508 standards for accessibility.

The following list includes the major accessibility features in IBM Security Identity Manager:

- Support for the Freedom Scientific JAWS screen reader application
- Keyboard-only operation
- Interfaces that are commonly used by screen readers
- Keys that are discernible by touch but do not activate just by touching them
- Industry-standard devices for ports and connectors
- The attachment of alternate input and output devices


**Keyboard navigation**

This product uses standard Microsoft Windows navigation keys.

**Related accessibility information**

You can view the publications for IBM Security Identity Manager in Adobe Portable Document Format (PDF) with the Adobe Acrobat Reader. The PDFs are available in the information center.

The following keyboard navigation and accessibility features are available in the form designer:

- You can use the tab keys and arrow keys to move between the user interface controls.
- You can use the Home, End, Page Up, and Page Down keys for additional navigation.
- You can start any applet, such as the form designer applet, in a separate window. The applet enables Alt+Tab to toggle between that applet and the web interface and for more screen workspace. To start the window, click Launch as a separate window.
- You can change the appearance of applets such as the form designer by using themes. Themes provide high contrast color schemes that help users with vision impairments to differentiate between controls.
IBM and accessibility

See the IBM Human Ability and Accessibility Center for information about the IBM commitment to accessibility.
Chapter 2. Application extensions

Application extensions can be defined in Java™ class files and run from a workflow.

Application extensions are typically defined in one or more Java class files. They are used when a set of functions needs to be run from a workflow. The functions can be implemented to receive input parameters from a workflow and return parameters back to the workflow.

WorkflowApplication interface

Application extensions that require access to the current workflow context information must implement the WorkflowApplication interface.

If the extension does not require any workflow context information, implementing this interface is not required. The following example is a code snippet for implementing the WorkflowApplication interface. For a complete example, see the information in the extensions directory.

```java
public class CustomEmail implements WorkflowApplication {
    public CustomEmail() {
    }

    When you implement the WorkflowApplication interface you must define a setContext method that accepts a WorkflowExecutionContext object. Store this object in a member variable in the implementing class.

    // The context of the workflow. Passed in from the workflow engine
    protected WorkflowExecutionContext ctx;

    /**
     * Passes the workflow execution context to the application.
     *
     * @param context WorkflowExecutionContext holding information about the
     * currently executing activity.
     */
    public void setContext(WorkflowExecutionContext ctx) {
        this.ctx = ctx;
    }
}
```

Application extension methods

The application can contain whatever processing is required to accomplish the task. An extension can contain any number of methods that can be exposed to the workflow.

The following example is a code snippet of a method that is available in the workflow for the extension node. For a complete example, see the information in the extensions directory.

```java
/**
 * Method sendMailByProperty.
 * This method is called to send an e-mail to an e-mail address specified by the
 * "recipient" property in the specified property file.
 * @param person - the requestee's person object
 * @param mailTag - the mailtag for this message. Used to look up properties
 * @param propertyFileName - the name of the property file that contains
```
* this message's data
* @param attrList - an optional list of tag/value pairs
* @return ActivityResult - a workflow activity result object
 */
public ActivityResult sendMailByProperty(Person person,
String mailTag,
String propertyFileName,
String attrs) {
String recipient_email = "";
try {
processSendMail(person,mailTag,propertyFileName,recipient_email,
attrs);
return new ActivityResult(ActivityResult.STATUS_COMPLETE,
ActivityResult.SUCCESS,
"Sent Mail",
null);
} catch (CustomEMailDataException e) {
return new ActivityResult(ActivityResult.STATUS_COMPLETE,
ActivityResult.FAILED,
e.getMessage(),
null);
}
}

Application Extension methods can receive inputs from the workflow. The inputs
defined in the workflow extension window maps to the method arguments (ensure
that the types match). The sendMailByProperty method returns an ActivityResult
object. This method allows the application to communicate back to the caller a
status and a return value, if necessary. The ActivityResult object has member
variables for status (int), summary, (String), detail (List), and description (String).
Return values are in the detail list. The order of the values in the list must
correspond to the order of the output parameters as defined in the extension
window. See the IBM Security Identity Manager API documentation for a complete
description of the ActivityResult class.

Registering extensions

For the workflow to make the extension available with the extension node, it must
first be registered in the ISIM_HOME/data/workflowextensions.xml file.

Each method requires an activity entry in the XML file. The activity entry includes:

Activity ID
An activity ID is required and must be unique in the workflow. This name
is in the extension window activity type menu.

Implementation type
The implementation type contains the class name and the method name
that is started by this extension.

Parameters sections
The parameters sections lists the input and output parameters and their data
types. These parameters are in the extension window Input/Output
Parameters.

Transition restriction
The transition restriction defines the join type. Split type can also be
defined. For more information, see the information in the extensions
directory.

<ACTIVITY ACTIVITYID="SendMailByProperty" LIMIT="600000">
<IMPLEMENTATION_TYPE>
<APPLICATION...
The Application Extension class file must be in a JAR file, which must be in the IBM Security Identity Manager class path. After these changes are completed, you must restart the server before the extensions are available in the workflow.
Chapter 3. Application programming interfaces

Application programming interfaces (APIs) are part of a plug-in model that you can use to add applications without disrupting existing applications.

Remote application programs run outside of the IBM Security Identity Manager server Java virtual machine (JVM). Classes outside of the application packages are not intended to be started by a remote application. Classes in remote applications are documented under the IBM Security Identity Manager application packages. Server extensions, which run in the IBM Security Identity Manager server JVM, can use any of the classes listed in the published API documentation (Javadoc). They are Java classes that run in the same JVM of the caller. These APIs are used to develop IBM Security Identity Manager customization and extensions that can plug into IBM Security Identity Manager.

Several application APIs can be started by a remote application. A few server extension APIs in the dataservices package are included also. The following application APIs are intended to be started by a remote application:

**Provisioning Policy API**
Can search, add, modify, and delete provisioning policies in IBM Security Identity Manager from a remote application.

**Group API**
Can search, add, modify, and delete an IBM Security Identity Manager group.

**ACI API**
Can search, add, modify, and delete an access control information list (access right), but it does not verify authorization.

**Reconciliation API**
Can get, add, modify, and delete a reconciliation schedule for a particular service and triggers reconciliation.

The following server extension APIs are included:
- `com.ibm.itim.common.ComplexAttributeValue`
- `com.ibm.itim.dataservices.model.ComplexAttributeHandler`
- `com.ibm.itim.dataservices.model.domain.access.Access`
- `com.ibm.itim.dataservices.model.domain.access.ProvisioningConfiguration`
- `com.ibm.itim.dataservices.model.domain.access.NotificationOption`

**Applications API**

Use the applications API to create customized or alternative user interfaces. This API provides an interface to the IBM Security Identity Manager provisioning platform.

The applications API provides a set of Java classes that abstract the more frequently used functions of the provisioning platform. Examples are identity management, password management, and account management. The classes that make up this API are the same classes that IBM Security Identity Manager uses for its user interface.
For more information, see the documentation provided with the Applications API in the ISIM_HOME/extensions/version number/doc/applications directory. For sample codes, see the ISIM_HOME/extensions/version number/examples/apps directory.

Version number represents the version of IBM Security Identity Manager. For example:

ISIM_HOME/extensions/6.0/doc/applications
ISIM_HOME/extensions/6.0/examples/apps

**Self registration API**

Part of the applications API, the self registration API provides an interface to create a person in the provisioning platform without a user context.

The self registration API can be called without a user context. It is set up to start without accessing the system with an IBM Security Identity Manager account login and password. The Self Registration API is part of a customizable process. The process provides an example JavaServer Pages (JSP) page as a product extension based on the default inetOrgPerson class. The JSP calls the Self Registration API to create a user.

**Access control information list (ACI) API**

The ACI API provides an interface for managing the IBM Security Identity Manager access control list, container-by-container.

A remote client can use basic add, list, modify, and delete operations for managing the access control list. However, the ACI API cannot verify authorization to the user.

This API exists in the com.ibm.itim.apps.acl.AccessControlListManager class.

**IBM Security Identity Manager group API**

The IBM Security Identity Manager group API provides system group management capabilities, namely APIs to manage groups on the IBM Security Identity Manager service and groups on managed services. The APIs also provide search capabilities for these groups.

The IBM Security Identity Manager group API provides an interface for managing the groups on either the IBM Security Identity Manager service or on other managed services. You can search, add, modify, or delete these groups. You can also add and remove users in a group on either the IBM Security Identity Manager service or on a managed service.

For groups on the IBM Security Identity Manager service, the API exists in the following classes:

- com.ibm.itim.apps.system.SystemRoleManager
- com.ibm.itim.apps.system.SystemRoleMO
- com.ibm.itim.apps.system.SystemUserMO

For groups on a managed service, the API exists in the following classes:

- com.ibm.itim.apps.provisioning.GroupManager
- com.ibm.itim.apps.provisioning.GroupMO
Provisioning policy API

The IBM Security Identity Manager provisioning policy API provides an interface to manage provisioning policies that are defined in IBM Security Identity Manager.

This API can search, add, modify, and delete provisioning policy. The API exists in the following classes:
- com.ibm.itim.apps.policy.ProvisioningPolicyManager
- com.ibm.itim.apps.policy.ProvisioningPolicyMO

Recertification policy API

The IBM Security Identity Manager recertification policy API provides an interface to manage recertification policies that are defined in IBM Security Identity Manager.

This API provides capabilities to search, add, modify, delete, and run recertification policies.

The following classes or interfaces are exposed to provide recertification policy management capabilities through APIs.

1. Core classes:
   - com.ibm.itim.apps.policy.RecertificationPolicyManager
   - com.ibm.itim.apps.policy.RecertificationPolicyMO
   - com.ibm.itim.dataservices.model.policy.recert.RecertificationPolicy

2. Dependent classes:
   - com.ibm.itim.dataservices.model.policy.recert.RecertificationParticipant
   - com.ibm.itim.dataservices.model.policy.RoleTarget
   - com.ibm.itim.dataservices.model.policy.GroupTarget
   - com.ibm.itim.dataservices.model.policy.ServiceTarget
   - com.ibm.itim.scheduling.RecurringTimeSchedule

3. Abstract classes extended by recertification policy directly or indirectly:
   - com.ibm.itim.dataservices.model.policy.DirectoryPolicy
   - com.ibm.itim.dataservices.model.policy.ScopedPolicy
   - com.ibm.itim.dataservices.model.policy.ServicePolicy

4. Interface implemented by recertification policy or dependent classes directly or indirectly:
   - com.ibm.itim.dataservices.model.policy.Policy
   - com.ibm.itim.dataservices.model.policy.IPolicyTarget
   - com.ibm.itim.scheduling.Schedulable

Reconciliation API

The reconciliation API can create and query reconciliations and reconciliation parameters.

The Reconciliation API provides an interface to manage reconciliation schedules of services. You can:
- Get and set reconciliation schedules to a service.
- Modify the reconciliation schedules collection, which includes additions and deletions.
- Set the new collection.
- Trigger a specific reconciliation schedule to run.

The API exists in the following classes:
Authentication API

Use the authentication API for working with different trusted identity stores such as identity information. This information can be stored on a Windows domain server or an LDAP directory. It includes the use of different types of keys, typically passwords, to unlock the application for a user.

The authentication API contains the authentication client API, which makes authentication requests, and the authentication provider API, which implements authentication requests.

Data services API

The data services API provides an interface to the IBM Security Identity Manager data model.

This API abstracts the more commonly used data model entities such as identities, accounts, access, and services in the provisioning process. It includes a generic interface to handle complex attributes. Data synchronization depends on Data Services APIs. Furthermore, the data services API provides the data model that the Applications API uses.

Although the ability to change the data model is provided in this API, this ability is not its focus. The Data Services API is low level. It abstracts the physical layout of the data store (directory structure). It does not provide the business logic that the provisioning applications with the platform provide.

IBM Directory Integration API

With this API, IBM Tivoli® Directory Integrator can import identity information into IBM Security Identity Manager. It manages accounts in the IBM Security Identity Manager data store on external resources that use IBM Tivoli Directory Integrator.

The following features are included in this API:

**Note:** Directory Service Markup Language version 2 (DSMLv2) was deprecated.

- A Directory Service Markup Language version 2 (DSMLv2) ServiceProvider. You can use it to import data. IBM Security Identity Manager acts as a DSMLv2 client. IBM Tivoli Directory Integrator acts as a DSMLv2 server.
- A DSMLv2 event handler. You can use it to import data into IBM Security Identity Manager. IBM Security Identity Manager acts as a DSMLv2 server. IBM Tivoli Directory Integrator acts as a DSMLv2 client.
- Ready-to-use schema support for communicating with IBM Tivoli Directory Integrator. You can use IBM Tivoli Directory Integrator as an endpoint and define it as a service instance in the IBM Security Identity Manager user interface for identity feed.

JavaScript API

The JavaScript API extends the scripting components that are specific to the scripting language that is configured with the product.
IBM Security Identity Manager provides a method to register new JavaScript extensions with the server. You can use the JavaScript API to add additional objects and functions to the interpreter’s glossary. A client can create and register additional objects and functions with the interpreter to run at run time.

The JavaScript API provides information about access participants, such as participant type, workflow participants, group access management, and access notification context.

**Mail API**

Use the mail API to customize mail content, format, and notification recipients.

Clients who use this API can make notification requests and extend construction of notification messages. The Mail API contains the Mail Client API, which makes notification requests, and the Mail Provider API, which implements notification requests.

The mail API also contains a function that is called Post Office that prevents workflow participants from receiving multiple email notifications that have similar content. Similar emails are stored, combined into a single email notification, and forwarded to a user.

**Password rules API**

The password rules API provides an interface to customize the standard password rule set and random password generation process.

You can use the password rules framework to customize the mechanism of generating passwords by the IBM Security Identity Manager server. Use one of the following ways to add custom logic to the password framework:

- A custom rule
- A custom generator
- Custom rules and a custom generator

**Policy analysis API**

The policy analysis API provides an interface to information about policies that are defined in the IBM Security Identity Manager Server. It is an interface to the access granted to a specific individual.

The API contains a set of Java classes that retrieve and abstract the provisioning policy information that controls access to managed resources. The Provisioning Policy API reports the provisioning policy enforcement in the system, but it does not support client modification of the policy. A client can use the policy information for auditing or deciding about potential policy enforcement changes.

**Service provider API**

The service provider API provides custom connectors. The connectors can be used from the IBM Security Identity Manager provisioning platform or any other Java-based provisioning platform that supports the same interface.

Service provider APIs define the interface that the IBM Security Identity Manager adapter needs to implement and communicate to remote adapter agents. The
adapter agent implementation does not rely on IBM Security Identity Manager APIs except for the set of asynchronous notification APIs provided under Service Provider APIs.

The following operations are included in the interface between the provisioning platform and the connector:

- Add
- Change password
- Delete
- Modify
- Restore
- Search
- Suspend
- Test

The provisioning platform performs all of the operations needed to determine the actions and their parameters that are to be run against resources. The connector runs those operations on the resource within requirements that are related to the resource.

---

**Shared access APIs**

The shared access application programming interfaces are available with the shared access module.

**Shared access application APIs**

The Shared Access Application API provides an interface for shared access administration and shared access management.

- Shared Access Data Model APIs
  The Shared Access Data Model APIs are defined in the com.ibm.itim.dataservices.model.pim package.

  Supported APIs:
  - Credential
  - CredentialLease
  - CredentialPool
  - CredentialSetting
  - GlobalCredentialSetting
  - SAEntitlement
  - SAEntitlementCredentialObject
  - SAEntitlementCriteria
  - SAMembership
  - SharedAccessPolicy

  The SharedAccessSearchFilter class in com.ibm.itim.pim.authority package is used to define search filter for shared access search.

  For more information, see the API Javadoc in the ISIM_HOME/extensions directory.

- Credential vault administration APIs
  This set of APIs supports credential vault administration tasks:
  - Searching for credentials in the vault
  - Adding credentials to the vault
  - Getting and modifying credential settings
  - Registering passwords
- Getting historical passwords
- Getting current passwords
- Deleting credentials from the vault

The com.ibm.itim.apps.pim package provides the following classes to support the tasks:

CredentialManager
CredentialMO

- Credential pool administration
  This set of APIs supports credential pool administration tasks:
  - Searching for credential pools
  - Adding credential pools
  - Getting and modifying the credential pool configuration
  - Deleting credential pools

The com.ibm.itim.apps.pim package provides the following classes to support the tasks:

CredentialPoolManager
CredentialPoolMO

- Shared access policy administration
  This set of APIs supports shared access policy administration tasks:
  - Searching for shared access policies
  - Adding shared access policies
  - Getting and modifying shared access policies
  - Deleting shared access policies

The com.ibm.itim.apps.pim package provides the following classes to support the tasks:

SharedAccessPolicyManager
SharedAccessPolicyMO

- Shared access batch load
  The SABatchLoadManager class in the com.ibm.itim.apps.pim package supports shared access batch load.

- Shared access management
  This set of APIs supports shared access management tasks:
  - Searching for authorized shared access for a user
  - Searching for available shared access in the system
  - Searching for authorized roles for shared access
  - Searching for shared access entitlements by role
  - Checking whether shared access checkout is authorized for a user
  - Checking whether viewing of the credential password is authorized for a user
  - Checking out a credential
  - Checking in a credential
  - Returning credential lease information

The com.ibm.itim.apps.pim package provides the following classes to support the tasks:

SharedAccessManager
CredentialLeaseManager
CredentialLeaseMO
Shared access web services API

The WSSharedAccessService web service supports shared access management functions for the IBM Security Identity Manager shared access module.

The methods support the following operations:
- Logging in and logging out
- Obtaining shared access credentials and credential attributes
- Check in of individual credentials
- Check in of all checked out credentials
- Check out of credentials, with or without an accompanying workflow
- Searching for shared access for a specified service. The search can be based on either the unique resource identifier of the service or on the Distinguished Name of the service.

For more information, see ISIM_HOME/extensions/product_version_number/doc/ws/ITIMWS.odt.

For examples of usage of the API, see ISIM_HOME/extensions/product_version_number/examples/ws/Readme.html.

Shared access authorization extension APIs

You can extend the shared access authorization engine by creating plug-in custom authorization providers.

The com.ibm.itim.pim.authority package provides a SharedAccessAuthorizationStrategy interface that you can use to implement the plug-ins.

For an example of using shared access authorization extension, see ISIM_HOME/extensions/product_version_number/examples/sa_authority/Readme.

Shared access JavaScript APIs

Shared access provides the CredentialModelExtension API to support access to a Credential object in various shared access-related workflow operations.

You can use the access to customize operations, including addCredentialToVault, checkin, and checkout.

For more information, see "CredentialModelExtension" on page 56.

Single sign-on API

The single sign-on API provides a single sign-on interface to accessible resources.

Some IBM Security Identity Manager installations might require integration with third party, single sign-on providers. Typically, such single sign-on providers protect a set of web-based resources with an authentication data store that is managed separately from IBM Security Identity Manager. The first time a client attempts to access any protected resource, the single sign-on provider provides authentication. If access is granted, the provider passes a token that indicates the identity of the authenticated user to all resources that are accessed later.
Web services API

This API consists of multiple web services, which are grouped by function. The services are listed alphabetically except the WSSessionService. This service is listed first since it is the first service that is called by any application. The session object that is returned by its login method is used as a parameter in all subsequent services.

**WSSessionService**

The WSSessionService web service provides authentication, session creation, and password challenge authentication. A client calls WSSessionService before you start any other web services. WSSessionService returns a session (handle) object that must be passed to the other web service calls to maintain a threaded conversation. The service provides the following operations:

- Login.
- Logout.

You can also use the WSUnauthService web service for other operations.

**WSAccessService**

The WSAccessService web service provides the following operations:

- Create a user access.
- Retrieve existing user access of a person.
- Remove user access.
- Search access entitlements available to a person.

The service provides following operations:

- Create and modify accesses.
- Do access searches.

**WSAccountService**

The WSAccountService web service provides the following operations to do account-related tasks:

- Create, modify, and other simple account operations.
- Retrieve default account attributes for a new account as specified by the provisioning policy.
- Retrieve the account profile name for a service.

**WSExtensionService**

The WSExtensionService web service provides a framework to extend the existing web services that are used by users. The service provides the users to create an operation to show a new IBM Security Identity Manager API. The detailed steps to create an extension service are specified in the ITIMWS.odt file, which is in the ISIM_INSTALL_DIR/extensions/6.0/doc/ws directory. ISIM_INSTALL_DIR is the directory where IBM Security Identity Manager is installed.

**WSGroupService**

The WSGroupService web service provides group management functions. The service provides the following operations:
• Create and remove groups.
• Search groups.
• Manage group membership.

**WSOrganizationalContainerService**

The `WSOrganizationalContainerService` web service provides IBM Security Identity Manager organization tree traversal and retrieval methods.

**WSPasswordService**

The `WSPasswordService` web service provides password management functions. The service provides the following operations:
• Validates the password as per the password policy rules.
• Enables change or generate password.

**WSPersonService**

The `WSPersonService` web service provides person-object related methods. The service provides the following operations:
• Create, modify, suspend, restore, delete, and other simple person operations.
• Retrieve the services to which a person is entitled in IBM Security Identity Manager or accounts.
• Do person searches.
• Retrieve the person object of the Principal.

**WSProvisioningPolicyService**

The `WSProvisioningPolicyService` web service deals with the provisioning policy. The service provides the following operations:
• Search provisioning policies.
• Create, modify, and delete provisioning policies.

**WSRequestService**

The `WSRequestService` web service provides the IBM Security Identity Manager request related functions. The service provides the following operations:
• Search for completed requests.
• Retrieve pending requests.
• Retrieve the request object that is based on the process ID or request ID.

**WSRoleService**

The `WSRoleService` web service provides role-based capabilities in the IBM Security Identity Manager. The service provides the following operations:
• Create and modify roles.
• Do role searches.
• Manage role hierarchy.
**WSSearchDataService**

The WSSearchDataService web service provides functions to search various IBM Security Identity Manager directory objects. The search method does not enforce the IBM Security Identity Manager ACIs, but a valid IBM Security Identity Manager session is required to call these methods. The service provides the following operations:

- Search for persons from root container.
- Search for persons that are having an IBM Security Identity Manager account.
- Search for the possible delegates within IBM Security Identity Manager for the logged-in user.
- Retrieve the searchable attributes of an entity in IBM Security Identity Manager.
- Retrieve common searchable attributes for the IBM Security Identity Manager entity.

**WSServiceService**

The WSServiceService web service provides IBM Security Identity Manager-based managed services (end-point configuration) functions. The service provides the following operations:

- Retrieve support data. For example, group data for UNIX, Linux, or Microsoft Windows services.
- Determine whether a password is required when provisioning on a service.
- Retrieve services that are configured on IBM Security Identity Manager.

**WSSharedAccessService**

The WSSharedAccessService web service provides many functions for the shared access module that is introduced in IBM Security Identity Manager Version 6.0. The web service clients must call the login method before it calls any other web services. The service provides the following operations:

- Retrieve authorized shared accesses.
- Retrieve the credentials.
- Check in or checkout credentials.

**Note:** You must install and enable the shared access module in order to use the WSSharedAccessService API. For more information, see “Shared access web services API” on page 14.

**WSSystemUserService**

The WSSystemUserService web service provides the functions that are related to system users. The service provides the following operations:

- Manage delegates, that is, add, modify, or delete delegates.
- Retrieve all the system roles.
- Configure challenge response.
- Search for system users who have an IBM Security Identity Manager account.
WSToDoService

The WSToDoService web service provides the functions to manage the different activities available in IBM Security Identity Manager. The service provides the following operations:

- Approve or reject activities.
- Retrieve or Submit Request for information activity details.
- Retrieve the pending activities of the logged-in user.

WSUnauthService

The WSUnauthService web service provides an interface for all the web service APIs that do not require the IBM Security Identity Manager authentication. The service provides the following operations:

- Version information.
- Reset password by using the challenge responses.
- Password policies.

Workflow API

Use the workflow API for custom code that can be called from a workflow process as a custom Java application or a JavaScript function. This custom code can then do special business logic, query external data stores, or provide integration with other workflow engines.

The Workflow API consists of a set of Java classes. The classes abstract the more commonly used concepts of the workflow environment, such as processes, activities, and relevant data.

The Workflow API supports new access request types. The access owner is a participant type.

The Workflow API provides methods for updating the recertification state and provides audit information for recertification. Audit records contain information about the recertification configuration and the who, what, and when of recertification tasks. These audits provide more useful reports about recertification compliance of users, accounts, and accesses. Consumers of the recertification policies can also have their recertification process audited in a reportable way.
Chapter 4. Dictionary for a password policy

You can create a dictionary for a password policy rule that rejects certain terms as passwords.

To use a dictionary for a password policy rule, you must first create and load a dictionary.ldif file to the IBM Security Identity Manager Server. To create a dictionary for a password policy rule:

1. Using an ASCII or other plain text editor, create a dictionary that contains the list of terms in an LDAP Data Interchange Format (LDIF) file.
   For example, create a file similar to this dictionary.ldif file, which specifies the domain as dc=com:
   
   \begin{verbatim}
   dn: erword=test,erdictionaryname=password, ou=itim, dc=com
   erWord: test
   objectclass: top
   objectclass: erDictionaryItem
   
   dn: erword=secret,erdictionaryname=password, ou=itim, dc=com
   erWord: secret
   objectclass: top
   objectclass: erDictionaryItem
   
   dn: erword=password,erdictionaryname=password, ou=itim, dc=com
   erWord: password
   objectclass: top
   objectclass: erDictionaryItem
   \end{verbatim}

2. Load the dictionary.ldif file on the IBM Security Identity Manager Server with one of these procedures:
   - Use an LDAP browser to import the dictionary.ldif file.
   - At a command prompt on the IBM Security Identity Manager Server, enter this LDAP command on one line:
     \begin{verbatim}
     ITDS_HOME/bin/ldapadd.exe -h hostname -D cn=adminuser -w adminpwd -V 3 -f dictionary.ldif
     \end{verbatim}
     
     -h \textit{hostname}  
     Specifies the host name of the computer on which the LDAP server is running.
     
     -D \textit{cn=adminuser}  
     Specifies the administrator's distinguished name to bind to the LDAP directory.
     
     -w \textit{adminpwd}  
     Specifies the administrator's distinguished name password, for simple authentication.
     
     -V \textit{ldap_version}  
     Specifies the version of the LDAP protocol to use. The default value is 3, for the LDAP v3 protocol. A value of 2 uses the LDAP v2 protocol.
     
     -f \textit{file}  
     Reads the entry modification information from a file such as dictionary.ldif, instead of from standard input.

The dictionary file can now be used in the password strength rule.
Chapter 5. Dataservices attributes for recertification

IBM Security Identity Manager provides optional attributes in the erAccountItem object class to represent different values for recertification.

Overview

The dataservices attributes for recertification are relevant only if recertification is enabled for specific accounts or accesses.

The following optional attributes are provided:

- erLastCertifiedDate
- erRecertificationLastAction
- erAccessLastCertifiedDate
- erAccessRecertificationLastAction

**erLastCertifiedDate**

The erLastCertifiedDate attribute is updated by the account recertification process only, but not for accesses. An optional attribute for the timestamp of the last time the account was marked as recertified. This attribute is updated on approved recertifications regardless of recertification policy schedule type, whether rolling or calendar style.

This attribute is updated for both approvals during normal recertification cycle and through the recertificationOverride option outside of the normal recertification policy run. The absence of a value means that recertification was never approved for this account. The Account data services object from the com.ibm.itim.dataservices.model.domain package defines the setLastCertifiedDate() and getLastCertifiedDate() methods for accessing this attribute. When an account is certified, this attribute must be updated along with erRecertificationLastAction.

**erRecertificationLastAction**

The erRecertificationLastAction attribute is updated by the account recertification process only, but not for accesses. This attribute requires a getter and setter method defined on the Account data services object class com.ibm.itim.dataservices.model.domain package:

```java
public void setRecertificationLastAction(String recertificationAction)
public String getRecertificationLastAction()
```

This optional attribute describes the action taken the last time recertification was run. The following values are valid:

- com.ibm.itim.dataservices.model.domain.Account.CERTIFIED = 'CERTIFIED'
- com.ibm.itim.dataservices.model.domain.Account.CERTIFIED_ADMIN = 'CERTIFIED_ADMIN'
- com.ibm.itim.dataservices.model.domain.Account.REJECTED_MARK = 'REJECTED_MARK'
- com.ibm.itim.dataservices.model.domain.Account.REJECTED_SUSPEND = 'REJECTED_SUSPEND'
erAccessLastCertifiedDate

The erAccessLastCertifiedDate attribute is specific to accesses that are defined on an account. This multivalued attribute holds the access group definition distinguished name and timestamp that shows when that access was last certified as a delimited string.

Example

```
eraccesslastcertifieddate: ernntlocalname=users, erglobalid=7281584268561021074, ou=services, erglobalid=00000000000000000000, ou=hawk,o=ibm, c=us;;200711202115Z
```

This example shows the last recertification date for the access that is associated with the access defined for the group specified by the distinguished name. Only one value for this attribute per access is defined for the account.

erAccessRecertificationLastAction

The erAccessRecertificationLastAction attribute is specific to recertification state of accesses that are defined on an account. This multivalued attribute holds the access group definition distinguished name and recertification last action taken as a delimited string. It serves the same purpose for accesses as erRecertificationLastAction does for accounts.

Example

```
eraccessrecertificationlastaction: ernntlocalname=users, erglobalid=7281584268561021074, ou=services,erglobalid=00000000000000000000, ou=hawk,o=ibm, c=us;;CERTIFIED
```

This example shows the last recertification action for the access that is associated with the group definition distinguished name. The values for the action are the same as described for the erRecertificationLastAction attribute. Only one value for this attribute per access is defined for the account.
Chapter 6. Workflow extensions for recertification

IBM Security Identity Manager provides workflow extensions for tracking the recertification state.

Overview

A recertification extension is code that can be called directly from a workflow. An extension defined for accounts also handles the recertification state for accesses, and uses dataservices to update attributes stored on the account object in data services. These extension methods are integrated into the AccountExtensions class from the com.ibm.itim.workflowextensions package.

Because the recertification extensions provided are considered activities by the workflow engine, any failure in those extensions is returned as a failure when the activity completes. This result causes the recertification workflow to fail, and its failure is audited in the RECERTIFICATIONLOG audit table as well.

The following extensions are provided:
- recertificationMark
- recertificationMarkAccess
- recertificationSuspend
- recertificationCertify
- recertificationCertifyAccess
- recertificationAdminCertify
- recertificationAdminCertifyAccess

recertificationMark

The public ProcessResult recertificationMark(Account) extension updates erLastRecertificationAction for the target type, updating the erLastRecertificationAction attribute to:

```
com.ibm.itim.dataservices.model.domain.Account.REJECTED_MARK = 'REJECTED_MARK'
```

The recertification action is audited in RECERTIFICATIONLOG table for use by reports.

recertificationMarkAccess

The public ProcessResult recertificationMark(UserAccessAccount) extension has the same function for accesses as recertificationMark() has for users and accounts. It updates the erAccessLastRecertificationAction attribute specific to the UserAccess passed in to:

```
com.ibm.itim.dataservices.model.domain.Account.REJECTED_MARK = 'REJECTED_MARK'
```

The recertification action is audited in RECERTIFICATIONLOG table for use by reports.

Note: This method is for suspending accounts only. No method for suspending access is provided.
**recertificationSuspend**

The public ProcessResult `recertificationSuspend(Account)` extension updates `erLastRecertificationAction` for the account. It updates the `erLastRecertificationAction` attribute to:

```
com.ibm.itim.dataservices.model.domain.Account.REJECTED_SUSPEND = 'REJECTED_SUSPEND'
```

The recertification action is audited in RECERTIFICATIONLOG table for use by reports.

**Note:** This method is for suspending accounts only. No method for suspending access is provided.

**recertificationCertify**

The public ProcessResult `recertificationCertify(Account)` extension updates `erLastRecertificationAction` for the target type. It updates the `erLastRecertificationAction` attribute to:

```
com.ibm.itim.dataservices.model.domain.Account.CERTIFIED = 'CERTIFIED'
```

The recertification action is audited in RECERTIFICATIONLOG table for use by reports. This extension also updates the `erLastCertifiedDate` attribute with the current timestamp.

**recertificationCertifyAccess**

The public ProcessResult `recertificationCertify(UserAccessAccount)` extension updates `erLastAccessRecertificationAction` for the access. It updates the `erLastRecertificationAction` attribute for the specified `UserAccess` to:

```
com.ibm.itim.dataservices.model.domain.Account.CERTIFIED = 'CERTIFIED'
```

The recertification action is audited in RECERTIFICATIONLOG table for use by reports. This extension also updates the `erAccessLastCertifiedDate` attribute for the `accessAttribute` with the current timestamp.

**Note:** This method is the access version of `recertificationCertify` for users and accounts.

**recertificationAdminCertify**

The public ProcessResult `recertificationAdminCertify(Account)` extension updates `erLastRecertificationAction` for the target type. It updates the `erLastRecertificationAction` attribute to:

```
com.ibm.itim.dataservices.model.domain.Account.CERTIFIED_ADMIN = 'CERTIFIED_ADMIN'
```

The recertification action is audited in RECERTIFICATIONLOG table for use by reports. This extension also updates the `erLastCertifiedDate` attribute with the current timestamp.

**recertificationAdminCertifyAccess**

The public ProcessResult `recertificationAdminCertify(UserAccessAccount)` extension updates `erLastAccessRecertificationAction` for the access. It updates the `erAccessLastRecertificationAction` attribute for the `UserAccess` passed in to:

```
com.ibm.itim.dataservices.model.domain.Account.CERTIFIED_ADMIN = 'CERTIFIED_ADMIN'
```
The recertification action is audited in RECERTIFICATIONLOG table for use by reports. This extension also updates the erAccessLastCertifiedDate attribute for the accessAttribute with the current timestamp.

**Note:** This method is the access version of recertificationAdminCertify for users and accounts.
Chapter 7. Date range customization

IBM Security Identity Manager provides additional date range customization, which is not available through the standard Form Designer applet.

With these options, you can control the years available to users when they customize a date. The following options must be configured manually on the following form template that is stored in the directory server:

```
erformname=inetOrgPerson,ou=formTemplates,ou=itim,ou=tivsys,dc=com
```

You can specify options that define the range of years to be displayed. You also can specify the standard range of years, a special extended max year such as 9999, or special minimum value such as 1900. You have options to display all years between the standard range and extended dates.

The options are:

- **minYear**
  Minimum year to display.

- **spanMinYearRange**
  When set to a value of false, displays all years between minYear and minRangeYear.

- **minRangeYear**
  Starting year for the standard range of years. The default is 1990.

- **maxRangeYear**
  Ending year for the standard range of years. The default is 2010.

- **spanYearRange**
  When the value is false, displays all years between maxRangeYear and maxYear.

- **maxYear**
  Maximum year to display.
Chapter 8. Dynamic tags in mail templates

IBM Security Identity Manager mail templates allow dynamic retrieval, substitution, and decision making in creating a message.

Dynamic content tags and examples

IBM Security Identity Manager provides dynamic content tags to allow text substitution and enable translation. The tags are used for the emails generated by these tasks:

- Designing workflows
- Specifying mail activity
- Manual service notification
- Recertification notification
- Post office
- Reminder template
- Default system notifications
- Delegation notifications

The tags include:

**JavaScript code**
Handles JavaScript and runs the JavaScript content contained between the open and close tags. This tag contains child tags unless they return a string. JavaScript code is called in `<JS>MyJavaScriptCode</JS>` delimiters.

Table 1. Syntax and example of using JavaScript code to replace message content

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;JS&gt;text or JavaScript tag&lt;/JS&gt;</code></td>
<td>Enter each <code>&lt;JS&gt;&lt;/JS&gt;</code> statement as a single line: An account request has been initiated for <code>&lt;JS&gt;process.requesteeName;&lt;/JS&gt;</code> <code>&lt;JS&gt;</code>if (var x=process.getParent() !=null) return x  ```&lt;/JS&gt;`</td>
</tr>
<tr>
<td><code>&lt;JS escapeentities=&quot;false&quot;&gt;text or JavaScript tag&lt;/JS&gt;</code></td>
<td>When specified as &quot;false&quot;, any text returned by the JavaScript execution does not have its HTML entity tags escaped. For instance, the <code>&lt;</code> character does not return as &lt;. This option might be useful when the execution of the JavaScript code returns XML. For example, embedding XHTML body notifications inside the XHTML body of the post office template. The default for this attribute is &quot;true&quot;, so not specifying the tag escapes the characters.</td>
</tr>
</tbody>
</table>
### Replace tag

Formats the message represented by the key to allow string replacement. The formatted string can have zero or more parameters. Parameters can contain strings, activity IDs, or JavaScript. The string inside the key must exist in the CustomLabels.properties file. Strings are sourced from a CustomLabels.properties resource bundle file or from the Labels.properties file.

The key of the string replacement can be specified with the key attribute or by adding a `KEY` tag between `RE` tags. Specifying a key that uses both the attribute and tag at the same time results in an exception.

The tag has these parameters:

- **Key tag**
  
  Represents the resource bundle key for a `RE` tag. For example:
  ```xml
  <RE key="key">
  </RE>
  ```

- **PARM tag**

  Represents the parameters for a `RE` tag. For example:
  ```xml
  <RE key="key">
  <PARM>with plain text</PARM>
  </RE>
  ```

### Table 1. Syntax and example of using JavaScript code to replace message content (continued)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
</table>
| `<JS removexhtmlheader="false">text or JavaScript tag</JS>` | If `removexhtmlheader="true"` is in the JS tag, any text returned from the JavaScript does not have the DTD statement in the XHTML content. The text returned from the JavaScript has the DTD statement in the XHTML content when either of the following conditions exist:  
  * `removexhtmlheader="false"`.  
  * It is not placed in the JS tag.  
  The default value of this attribute is `false`. Not specifying the flag in the tag puts the DTD statement in the XHTML content. |

### Table 2. Syntax and examples of using a `RE` tag to replace message content

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
</table>
| `<RE key="message key">`  
`<PARM>text or JavaScript tag</PARM>`  
`</RE>` | `<RE key="message key">`  
`<PARM>with plain text</PARM>`  
`<PARM>`process.requesteeName;`  
`<JS>`</JS>`<PARM>`<PARM>`</PARM>`
`</RE>` |
| or enter each `<KEY>`/`/KEY>` statement as a single line:  
`<RE>`<KEY>`message key or JavaScript tag to return a key`  
`</KEY>`  
`<PARM>text or JavaScript tag</PARM>`  
`</RE>` |  
Output:  
This is a formatted string replacement example `with plain text` and JavaScript code for requestee name `John Smith`. |

The `KEY` can be specified by either an attribute on the `RE` tag, or as a subelement of the `RE` tag by using the tag `KEY`. |
Table 2. Syntax and examples of using a RE tag to replace message content (continued)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>To enable string replacement for translation, specify a custom label in</td>
<td><code>&lt;RE key=&quot;readOnlyDateFormat&quot;&gt;</code>&lt;br&gt;<code>&lt;PARM&gt;</code>&lt;br&gt;<code>&lt;JS&gt;</code>if (process.scheduled !=null)&lt;br&gt;return process.scheduled.getTime();&lt;br&gt;else&lt;br&gt;return &quot;&quot;;&lt;br&gt;<code>&lt;/JS&gt;</code>&lt;br&gt;<code>&lt;/PARM&gt;</code>&lt;br&gt;<code>&lt;/RE&gt;</code>&lt;br&gt;Output:&lt;br&gt;Apr 18, 2005 05:20:52 EDT</td>
</tr>
<tr>
<td>a CustomLabels.properties file to overwrite a Labels.properties key.</td>
<td></td>
</tr>
<tr>
<td>For example, the Labels.properties file contains this key/value pair:</td>
<td></td>
</tr>
<tr>
<td>readOnlyDateFormat=MMM dd, yyy hh:mm:ss z</td>
<td></td>
</tr>
<tr>
<td>To override this format, add the same key to the CustomLabels.properties</td>
<td></td>
</tr>
<tr>
<td>file.</td>
<td></td>
</tr>
</tbody>
</table>

Non-compliant message tag
Represents a message that describes the noncompliant attributes of an account. For example:

`<CAMessage/>`

Dynamic content message tags
Tags are delimited in `<TAG/>` syntax, such as the following examples:

Table 3. Syntax and example of using tags to replace message content

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;TagName/&gt;</code></td>
<td><code>&lt;CAMessage/&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Returns a string that describes the non-compliant attributes of an account.</td>
</tr>
<tr>
<td></td>
<td><code>&lt;ManualServiceAddAccount/&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Returns a string that contains the text body for manual service email notification.</td>
</tr>
<tr>
<td></td>
<td><code>&lt;rfiActivityHasBeenSubmitted/&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Returns a string that contains the text body of an RFI activity that was submitted in an account request workflow.</td>
</tr>
</tbody>
</table>

ID tag
Represents the activity ID in the form: `Process.ActivityId`. For example:

`<ID/>`

ITIMURL tag
Based on group membership of the person. It represents the URL of the IBM Security Identity Manager Server. A forced URL can be applied by using the `forcedurl` attribute of the tag. This attribute contains constant values such as the value `console` or the value `enduser`.

Table 4. Syntax and examples of ITIMURL

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;ITIMURL/&gt;</code></td>
<td>Based on group membership of the person. It represents the URL of the IBM Security Identity Manager Server.</td>
</tr>
</tbody>
</table>
Table 4. Syntax and examples of ITIMURL (continued)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
</table>
| <ITIMURL forcedurl="enduser"/> | Represents the URL of the graphical user interface on the IBM Security Identity Manager Server. If the forcedurl attribute is used, the URL is not generated based on the group membership of the person.  
Values for this attribute include:  
enduser  
The URL points at the self-service graphical user interface.  
console  
The URL points at the administrator graphical user interface. |
| <ITIMURL forcedurl="console"/> | Represents the URL of the graphical user interface on the IBM Security Identity Manager Server. If the forcedurl attribute is used, the URL is not generated based on the group membership of the person.  
Values for this attribute include:  
enduser  
The URL points at the self-service graphical user interface.  
console  
The URL points at the administrator graphical user interface. |

Properties file values

To change templates, you can add the key=value statements in the CustomLabels.properties file or create your own properties and values.

Required escape characters and JavaScript

The following characters must be escaped by using the appropriate HTML entity form that has the format &entity;. This action ensures that the notification template XML is well-formed.

Table 5. Escape characters

<table>
<thead>
<tr>
<th>Escape character</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less Than (&lt;)</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater Than (&gt;)</td>
</tr>
<tr>
<td>&amp;quote;</td>
<td>Quotation (&quot; )</td>
</tr>
<tr>
<td>'</td>
<td>Apostrophe (’ )</td>
</tr>
<tr>
<td>&amp;</td>
<td>Ampersand (&amp; )</td>
</tr>
</tbody>
</table>

For example, to use the following JavaScript:

```javascript
if (i<4) return "less than four";
```

The dynamic content tag is:
Common formatting patterns in the XHTML body

Default messages are formatted with a common pattern in the XHTML body and also contain message-unique statements.

For example, the XHTML for the to-do reminder template calls a common style sheet (the imperatives.css file) and logos. Message-unique statements are similar to the following ones:

```xml
<!-- Start of notification body -->
<textBody/>
<RE key="escalation_note"/> <escalationTime/>
</td>
</tr>
<!-- End of notification body -->
```

The following example shows a complete set of statements in an XHTML body:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
PUBLIC "/W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>$TITLE</title>
<meta content="text/html; charset=UTF-8" http-equiv="Content-Type" />
<link type="text/css" title="Styles" rel="stylesheet"
href="$BASE_URL/console/css/imperative.css" />
</head>

<!-- Put Next statement on one line -->
<body topmargin="0" marginheight="0" leftmargin="0" marginwidth="0"
bcolor="ffffff">

<!-- Block for the Template Header part -->
<table width="100%" border="0" cellpadding="0" cellspacing="0">
<tr>
<!-- Tivoli logo -->
<td width="186" background="$BASE_URL/console/html/images/mid-part-1.gif">
<img src="$BASE_URL/console/html/images/left-tiv-1.gif" alt="$LOGO_ALT" /></td>
<!-- Middle part -->
<td background="$BASE_URL/console/html/images/mid-part-1.gif" width="692"></td>
<!-- IBM logo -->
<td background="$BASE_URL/console/html/images/ibm_banner.gif" width="96"></td>
</tr>
</table>

<!-- Title Bar -->
<table width="100%" border="0" cellpadding="0" cellspacing="0">
<tr>
<!-- ITIM Notification Label -->
<td background="$BASE_URL/console/html/images/titlebar_middle.gif"
height="23" width="8">$TITLE</td>
</tr>
</table>
```

Chapter 8. Dynamic tags in mail templates
Mail templates

You define mail templates to deliver customized message notifications. The templates use several customization functions.

Templates have these main parts:

Subject
Describes an activity to a recipient of the notification. The subject can consist of plain text and dynamic content tags. If no subject is specified for manual service activities, no email is sent.

Text body
Describes the outcome of an activity, such as an account approval. The content can consist of plain text, dynamic content tags, and JavaScript code.

XHTML body
Provides the content of the email as an HTML message.

Dynamic content can include dynamic content message tags, JavaScript code, and tags that replace variables with other values or reference a property that allows translation with the CustomLabels.properties file.
Manual service default messages

IBM Security Identity Manager provides default notification templates for messages that participants that are service owners receive when changes occur to accounts or passwords for manual services that they own.

Default notification templates

IBM Security Identity Manager provides these default notification templates:

<ManualServiceAddAccount/>
Provides default text sent to a participant when an account is added for the user of a manual service.

<ManualServiceModifyAccount/>
Provides default text sent to a participant when an account is modified for the user of a manual service.

<ManualServiceDeleteAccount/>
Provides default text sent to a participant when an account is deleted for the user of a manual service.

<ManualServiceRestoreAccount/>
Provides default text sent to a participant when an account is restored for the user of a manual service.

<ManualServiceSuspendAccount/>
Provides default text sent to a participant when an account is suspended for the user of a manual service.

<ManualServiceChangePassword/>
Provides default text sent to a participant when a password change occurs for the user of a manual service.

Properties used for translation

If the properties exist in the CustomLabels.properties file, their value is used. Otherwise, the values of the properties in Labels.properties file are used. These properties contain the translated versions of the messages (with parameter substitution) that make up the dynamic tags. Change their values in the CustomLabels.properties file if you want different text. Do not change the defaults in the Labels.properties file.

The properties include these items:

manualServiceWorkOrderAddOperationMessage
manualServiceAttributeName
manualServiceAttributeValue
manualServiceAttributeAction
manualServiceAddAction
manualServiceRemoveAction
manualServiceReplaceAction
manualServiceWorkOrderChangePwdOperationMessage
manualServiceWorkOrderPwdValueMessage
manualServiceWorkOrderDeleteOperationMessage
manualServiceWorkOrderModifyOperationMessage
manualServiceWorkOrderRestoreOperationMessage
manualServiceWorkOrderSuspendOperationMessage
manualServiceUnknownPerson
Notification script example

A default notification script for a manual service provides a message that is sent to a participant. For example, the ManualServiceAddAccount notification output is similar to this example:

```
Attribute Name: Attribute Value
myattr: TT
Password: secret
Owner: Auditor
UserID: auditor1

Description: manual service operation
Requestee: Auditor
Subject: auditor1
Request Initiated: Jun 28, 2007 05:11:05 IST

Requested by process:
Process Name: Account Add
Description: Account Add Process
Requester: System Administrator
Requestee: Auditor
Subject: auditor1
```

Output example

The `<ManualServiceAddAccount/>` template provides a message that uses some of the values in the Labels.properties file:

```
manualServiceWorkOrderAddOperationMessage
manualServiceAttributeName : manualServiceAttributeValue
{insert real attribute names here} : {insert real attribute values here}
```

The `<ManualServiceModifyAccount/>` tag generates:

```
manualServiceWorkOrderModifyOperationMessage
{Place the following attributes on one line:}
manualServiceAttributeName : manualServiceAttributeValue
manualServiceAttributeAction
{insert real attribute names here} : {insert real attribute values here} :
{depending on what needs to be done, one of the following: }
{Place the following attributes on one line:}
{manualServiceAddAction,manualServiceReplaceAction,manualServiceRemoveAction}
```

Recertification default messages

IBM Security Identity Manager provides default message templates for recertification messages.

Default recertification templates

IBM Security Identity Manager provides default message templates for recertification messages. You cannot change the following templates:

Suspending Account

Provides default text that requests a participant to recertify use of an account. Declining the request suspends the account.

For example, the participant receives this message:

```
Recertification required for account myaccount on service shortword-linux
You have received a recertification request for account myaccount on service
```

Suspension:

```
ACCOUNT SUSPENDED
```

Recertification:

```
Recertification required for account myaccount on service shortword-linux
You have received a recertification request for account myaccount on service
```
shortword-linux owned by firstname lastname.

Rejection of this recertification request will result in the suspension of account myaccount on shortword-linux.

Activity: Recertification of Account/Access
Date submitted: Apr 26, 2007 10:34:51 IST
Request type: Recertification
Requested for: firstname lastname
Requested by: SYSTEM
Access/Account: myaccount
Description:

Due date: Apr 27, 2007 10:34:57 IST

Delete Account
Provides default text that requests a participant to recertify use of an account. Declining the request deletes the account.

For example, the participant receives this message:
Recertification required for account myaccount on service shortword-linux

You have received a recertification request for account myaccount on service shortword-linux owned by firstname lastname.

Rejection of this recertification request will result in the deletion of account myaccount on shortword-linux.

Activity: Recertification of Account/Access
Date submitted: Apr 26, 2007 10:34:51 IST
Request type: Recertification
Requested for: firstname lastname
Requested by: SYSTEM
Access/Account: myaccount
Description:
Due date: Apr 27, 2007 10:34:57 IST

Mark Account
Provides default text that is sent to a participant to recertify use of an account. Declining the request marks the account for a subsequent action on the account.

For example, the participant receives this message:
Recertification required for account myaccount on service shortword-linux

You have received a recertification request for account myaccount on service shortword-linux owned by firstname lastname.

Rejection of this recertification request will result in account myaccount on shortword-linux being marked as rejected for recertification.

Activity: Recertification of Account/Access
Date submitted: Apr 26, 2007 10:34:51 IST
Request type: Recertification
Requested for: firstname lastname
Requested by: SYSTEM
Access/Account: myaccount
Description:
Due date: Apr 27, 2007 10:34:57 IST

Mark Access
Provides default text that is sent to a participant to recertify use of an account on an access. Declining the request marks the access for a subsequent action on the account.
For example, the participant receives this message:
Recertification required for account myaccount on access myaccess.

You have received a recertification request for account myaccount on access myaccess owned by firstname lastname.

Rejection of this recertification request will result in access myaccess being marked as rejected for recertification.

Activity:
Date submitted: Apr 26, 2007 10:34:51 IST /* Need to fill this data */
Request type:
Requested for:
Requested by:
Access/Account:
Description:
Due date: Apr 27, 2007 10:34:57 IST

Delete Access
Provides default text that requests a participant to recertify use of an account on an access. Declining the request deletes the account on the access.

For example, the participant receives this message:
Recertification required for account myaccount on access myaccess.

You have received a recertification request for account myaccount on access myaccess owned by firstname lastname.

Rejection of this recertification request will result in the deletion of access myaccess.

Activity:
Date submitted: Apr 26, 2007 10:34:51 IST /* Need to fill this data */
Request type:
Requested for:
Requested by:
Access/Account:
Description:
Due date: Apr 27, 2007 10:34:57 IST

Account Suspended
Provides default text that is sent to a participant, confirming suspension of an account, after a participant declines a recertification request.

For example, the participant receives this message:
Account myaccount on service shortword-linux has been suspended due to rejection of a recertification request.

The account myaccount on service shortword-linux owned by firstname lastname has been suspended due to rejection of a recertification request.

Activity: Recertification of Account/Access
Date submitted: Apr 26, 2007 10:34:51 IST
Request type: Recertification
Requested for: firstname lastname
Requested by: SYSTEM
Access/Account: myaccount
Description:

Account Deleted
Provides default text that is sent to a participant, confirming deletion of an account, after a participant declines a recertification request.

For example, the participant receives this message:
Account myaccount on service shortword-linux has been deleted due to rejection of a recertification request.

The account myaccount on service shortword-linux owned by firstname lastname has been deleted due to rejection of a recertification request.

Activity: Recertification of Account/Access
Date submitted: Apr 26, 2007 10:34:51 IST
Request type: Recertification
Requested for: firstname lastname
Requested by: SYSTEM
Access/Account: myaccount
Description:
Due date: Apr 27, 2007 10:34:57 IST

Account Marked

Provides default text that is sent to a participant, confirming that an account is marked for suspension, after a participant declines a recertification request.

For example, the participant receives this message:

Account myaccount on service shortword-linux has been marked as rejected for recertification due to rejection of a recertification request.

The account myaccount on service shortword-linux owned by firstname lastname has been marked as rejected for recertification due to rejection of a recertification request.

Activity: Recertification of Account/Access
Date submitted: Apr 26, 2007 10:34:51 IST
Request type: Recertification
Requested for: firstname lastname
Requested by: SYSTEM
Access/Account: myaccount
Description:
Due date: Apr 27, 2007 10:34:57 IST

Access Marked

Provides default text that is sent to a participant. It confirms that an account on an access is marked for subsequent action after a participant declines a recertification request.

The template contains these statements:

Account myaccount on access myaccess has been deleted due to rejection of a recertification request.

The account myaccount on access myaccess owned by firstname lastname has been marked as rejected for recertification due to rejection of a recertification request.

Activity: Recertification of Account/Access
Date submitted: Apr 26, 2007 10:34:51 IST
Request type: Recertification
Requested for: firstname lastname
Requested by: SYSTEM
Access/Account: myaccount
Description:
Due date: Apr 27, 2007 10:34:57 IST

Access Removed

Provides default text that is sent to a participant, confirming deletion of an account on an access, after a participant declines a recertification request.

For example, the participant receives this message:
Account myaccount on access myaccess has been deleted due to rejection of a recertification request.

The account myaccount on access myaccess owned by firstname lastname has been deleted due to rejection of a recertification request.

Activity:
Date submitted: Apr 26, 2007 10:34:51 IST
Request type:
Requested for:
Requested by:
Access/Account:
Description:
Due date: Apr 27, 2007 10:34:57 IST

User Recertification Pending
Provides default text that is sent to a participant, confirming that a user recertification is pending, after a recertification request is initiated.

For example, the participant receives this message:
You have received a recertification request for user firstname lastname. The recertification includes their membership in X role(s) and ownership of Y account(s). Please indicate whether the user still requires these resources:

The account myaccount on access myaccess owned by firstname lastname has been deleted due to rejection of a recertification request.

Activity: Recertification of Account/Access/User
Date submitted: Sep 08, 2008 04:10:32 EDT
Request type: Recertification
Requested for: firstname lastname
Requested by: System
Due date: Sep 18, 2008 04:10:34 EDT

User Recertification Rejected
Provides default text that is sent to a participant, confirming that one or more resources were declined in a user recertification request.

For example, the participant receives this message:
One or more resources for user firstname lastname have been rejected during recertification.

The account myaccount on access myaccess owned by firstname lastname has been deleted due to rejection of a recertification request.

Activity: Recertification of Account/Access/User
Date submitted: Sep 08, 2008 06:30:07 EDT
Request type: Recertification
Requested for: firstname lastname
Requested by: System

The following roles were rejected:
rolename

The following accounts were rejected, along with all groups associated with the accounts:
Account "uid" on service "servicename"

The following groups were rejected, but the account was accepted:
Group "groupname" for account "uid" on service "servicename"
Properties file values

To change templates, you can use all of the key=value statements in the CustomLabels.properties file, or create your own properties and values.

The properties include these items on one line:

- `recertOn={0} on {1}`
- `recertTemplateSubject=Recertification required for account {0} on service {1}`
- `recertTemplateAccessSubject=Recertification required for account {0} on access {1}`
- `recertTemplateBody=You have received a recertification request for account {0} on service {1} owned by {2}.`
- `recertTemplateAccessBody=You have received a recertification request for account {0} on access {1} owned by {2}.`
- `recertDeclineSuspendsBody=Rejection of this recertification request will result in the suspension of account {0} on {1}.`
- `recertDeclineDeletesBody=Rejection of this recertification request will result in the deletion of account {0} on {1}.`
- `recertDeclineMarksBody=Rejection of this recertification request will result in account {0} on {1} being marked as rejected for recertification.`
- `recertDeclineDeletesAccessBody=Rejection of this recertification request will result in the deletion of access {0}.`
- `recertDeclineMarksAccessBody=Rejection of this recertification request will result in access {0} being marked as rejected for recertification.`
- `userRecertTemplateSubject=Recertification required for user {0}`
- `userRecertTemplateBody=You have received a recertification request for user {0}. The recertification includes their membership in {1} role(s) and ownership of {2} account(s). Please indicate whether the user still requires these resources.`
- `userRecertDeclineSubj=Recertification request rejected for user {0}`
- `userRecertDeclineBody=One or more resources for user {0} have been rejected during recertification. The following roles were rejected: {userRecertRolesRejectedLabel}. The following accounts were rejected, along with all groups associated with the accounts: {userRecertAccountsRejectedLabel}. The following groups were rejected, but the account was accepted: {userRecertGroupsRejectedLabel}. Account "{0}" on service "{1}" owned by "{2}"`
Recertification template key definitions

Recertification templates use the following key definitions:

- name=Activity
- timeScheduled=Date submitted
- recertRequestType=Request type
- recertRequestedFor=Requested for
- recertRequestedBy=Requested by
- recertAccountAccess=Access/Account
- recertDueDate=Due date
- recertRequestTypeName=Recertification

Workflow default messages

IBM Security Identity Manager provides default workflow messages.

Default workflow templates

All the workflow notice templates can be customized. IBM Security Identity Manager provides these default workflow notice templates:

Activity Timeout Template

Provides information that the workflow activity is timed out and terminated. By default, this template is enabled.

For example, the template provides this message:

Workflow activity is being timed out and will be terminated by the workflow system.

The following activity has timed out. The activity will be terminated by the workflow system and the result set to Terminated.

Activity Information

View Changes:http://localhost:9090/itim/console
Activity ID:ADApproval
Activity:AD Account Approval
Time Started:Jun 09, 2007 12:28:45 IST
Time Completed:
Result Summary:Escalated
State:Running
Activity Type:Manual Approval/Reject

Process Information

Process ID: 1099575082113388748
Activity:Default AD Account Approval Workflow
Description:
State:Running
Date submitted:Jun 09, 2007 12:23:41 IST
Time Completed:
Result Summary:
Requester:1099572462907357646
Requestee:firstname lastname
Subject:
Comment:
Detail:

The subject statement is:

<RE key="activity_timeout_subject" />

The plain text is:

<RE key="activity_timeout_message" />
<RE key="activity_timeout_detail" />
<RE key="activityInformation" />
Change Account Template

Provides information that the workflow activity has modified account information. By default, this template is disabled.

For example, the template provides this message:

Modified Account Information from IBM Tivoli Identity Manager

The following ITIM Service [ITIM] account has been modified:

View Changes:http://localhost:9090/itim/console
Process Reference:07501686186534505
Account ID:myaccount
Owner Name:firstname lastname
Time Completed:Jun 08, 2007 09:52:24 IST

The subject statement is:

<RE key="change_account_subject"/>

The plain text is:

<RE key="account_changed"><PARM>
<RE key="service_name_with_profile_name"><PARM>
<JS>EmailContext.getAccountServiceName();</JS></PARM>
<PARM><RE><KEY><JS>EmailContext.getAccountServiceProfileName();</JS></KEY></RE></PARM></RE></PARM></RE>

Chapter 8. Dynamic tags in mail templates 43
Compliance Template
Provides information that an account is not compliant with a provisioning policy. By default, this template is enabled.

For example, the template provides this message:
Compliance Alert for winlocal:
Account [helpdesk35] is not compliant with the provisioning policy.
The value [Performance Log Users] of attribute [Local Groups] should be [removed].

The subject statement is:

The plain text is:

Delegation Template
Provides the default template for delegation, which includes the new delegation information. By default, this template is enabled and cannot be disabled. If any exception is thrown while evaluating JavaScript in the notification template or parsing the notification template, then the default delegation notification is sent.

For example, the template provides this message:
You have been selected to be the delegate:

The subject statement is:

The plain text is:
Deprovision Account Template
Provides information that the workflow activity has removed an account.
By default, this template is enabled.
For example, the template provides this message:
Your account has been removed by IBM Tivoli Identity Manager.
The following Odessa Service [ADProfile] account has been deprovisioned.
View Changes: http://host:9080/itim/self
Process Reference:587039490436872731
Account ID:myaccount
Owner Name:myname
Reason:Policy Enforcement
Time completed:May 03, 2007 03:54:22 IST

The subject statement is:
<RE key="remove_account_subject"/>

The plain text is:
<RE key="account_deprovisioned">

<PARM><RE key="service_name_with_profile_name">
<PARM><JS>EmailContext.getAccountServiceName();</JS></PARM>
<PARM><RE><KEY><JS>EmailContext.getAccountServiceProfileName();</JS></KEY></RE></PARM></RE>

<ITIMURL/>
<RE key="processRef"/>: <JS>process.id;</JS>
<RE key="accountID"/>: <JS>EmailContext.getAccountUserId();</JS>
<RE key="accountOwnerName"/>: <JS>EmailContext.getAccountOwnerName();</JS>
<RE key="reason"/>: <JS>EmailContext.getReason();</JS>
<RE key="deprovisionCompleted"/>: <RE key="readOnlyDateFormat">
<PARM><JS>(new Date()).getTime();</JS></PARM></RE>

Manual Activity Approval Template
Provides information that the user should provide information for a request. By default, this template is enabled.
For example, the template provides this message:
Pending workflow action: Case 884089484067042.884089484067042
You have been requested to submit information for the following request
View Changes:http://localhost:9090/itim/console
Description:
Requestee:firstname lastname
Subject:subject
Request Initiated:Jun 08, 2007 10:27:29 IST
Process Reference:884089484067042
Requested by process:
Process ID:8840694840670432
Process Name:Provision Account
Description:Provisioning Account Process
Requester:System Administrator
Requestee:firstname lastname
Subject:subject

The subject statement is:
<RE key="pending_workitem_subject">"</PARM></RE>

The plain text is:
<RE key="wiApproval_message"/>
<ITIMURL/>
<RE key="description"/>: <RE><KEY><JS>process.description;</JS></KEY></RE>
<RE key="requestedFor"/>: <JS>process.requesteeName();</JS>
<RE key="subject"/>: <JS>process.subject();</JS>
<JS>if (process.subjectAccess!=null) if (process.subjectAccess.length>0)
{'<RE key="accessName"/>: <JS>process.subjectAccess();</JS>}</JS>
<JS>if (process.started != null) return process.started.getTime();"</PARM>

Chapter 8. Dynamic tags in mail templates 45
Manual Activity RFI Template

Provides the default template for request for information workflow activities. By default, this template is enabled

For example, the template provides this message:

You have been requested to submit information for the following request
Description:
Requestee: Shoe Flower
Subject: shoe1
Request Initiated: Aug 03, 2007 11:48:52 IST
Process Reference: 3053541330639294422
Requested by process:
Process ID: 3053541330639294422
Process Name: Provision Account
Description: Provision Account Process
Requester: System Administrator
Requestee: Shoe Flower
Subject: shoe1

The subject statement is:

<RE key="pending_workitem_subject" /><PARM><ID /></PARM></RE>

The plain text is:

<RE key="wiRFI_message" />
<ITIMURL/>
<RE key="description" />: <RE key="KEY">process.description</RE>
<RE key="requestedFor" />: <JS>process.requesteeName</JS>
<RE key="subject" />: <JS>process.subject</JS>
<JS>if (process.subjectAccess!=null)
if (process.subjectAccess.length>0)
{ '<RE key="accessName"/>
[ '<RE key="readOnlyDateFormat"><PARM>
<JS>if (process.started != null) return process.started.getTime();
else return '';</JS></PARM></RE>
<RE key="requestInit" />
<JS>if (process.started != null) return process.started.getTime();
else return '';</JS></PARM></RE>
<RE key="processRef" />
<JS>if (process.parentId == '0')
{ '<RE key="requestedBy"/>
<JS>process.requestorName</JS>
'; }
<JS>if (process.parentId == '0')
{ '<RE key="parent_process"/>
<JS>process.id</JS>
'; }
<JS>if (process.parentId != '0')
{ '<RE key="processID"/>
<JS>process.parentId</JS>
'; }
<JS>if (process.parentId != '0')
{ '<RE key="processName"/>
<JS>process.getParent().name</JS>
'; }
<JS>if (process.parentId != '0')
{ '<RE key="description"/>
<JS>process.getParent().description</JS>
'; }
<JS>if (process.parentId != '0')
{ '<RE key="requester"/>
<JS>process.getParent().requestorName</JS>
'; }
<JS>if (process.parentId != '0')
{ '<RE key="requestedFor"/>
<JS>process.getParent().requesteeName</JS>
'; }
<JS>if (process.parentId != '0')
{ '<RE key="subject"/>
<JS>process.getParent().subject</JS>
'; }
</JS>
Manual Activity Work Order Template

Provides default template for the workorder workflow manual activity. By default, this template is enabled.

For example, the template provides this message:

Pending workflow action:
Case 14019336458803275.1402011582339065124

You have received a Work Order request
The subject statement is:
<RE key="pending_workitem_subject" /><PARM><ID /></PARM></RE>

The plain text is:
<RE key="wiWorkOrder_message" />

New Account Template

Provides information that the workflow activity has created a new account. By default, this template is enabled.

For example, the template provides this message:

New Account Information from IBM Tivoli Identity Manager

The following new ITIM Service [ITIM] account has been created for you:

View Changes:http://localhost:80/itim/console
Process Reference:8498649245880216244
Password:bAMI#gai
Account ID:myaccount
Owner Name:firstname lastname

The subject statement is:
<RE key="new_account_subject" />

The plain text is:
<RE key="account_created" /><PARM><RE key="service_name_with_profile_name">
<PARM><JS>EmailContext.getAccountServiceName();</JS></PARM>
<PARM><JS>EmailContext.getAccountServiceProfileName();</JS></PARM></RE>
<ITIMURL/>
<RE key="processRef" />:<JS>process.id;</JS>
<JS><KEY><JS>process.getTransactionId() != '0' ?
  ['<RE key="TRANSACTION_ID_LABEL" />
  + EmailContext.getTransactionId(); ] : '</JS>
<RE key="password" />:<JS>EmailContext.getAccountPassword();</JS>
<RE key="accountID" />:<JS>EmailContext.getAccountUserId();</JS>
<RE key="accountOwnerName" />:
<JS>EmailContext.getAccountOwnerName();</JS>
<RE key="timeofprovision" />:<JS>(new Date()).getTime();</JS>
<JS><KEY><JS>EmailContext.hasNewAccess() ?
  ['<RE key="accountNewAccess" />
  + EmailContext.getAccountNewAccessAsString(); ] : '</JS>
<JS>EmailContext.getPasswordRetrievalUrl();</JS>
<JS><KEY><JS>EmailContext.getTransactionId() != '0' ?
  ['<RE key="RETRIEVE_PASSWORD_TITLE" />
  + EmailContext.getPasswordRetrievalUrl(); ] : '</JS>
<JS><KEY><JS>EmailContext.getTransactionId() != '0' ?
  ['<RE key="READONLYDATE_FORMAT" />
  + EmailContext.getTime(); ] : '</JS>
<KEY><JS>EmailContext.getTransactionId() != '0' ?
  ['<RE key="TRANSACTION_ID_LABEL" />
  + EmailContext.getTransactionId(); ] : '</JS>
<KEY><JS>EmailContext.getTime(); </JS>
New Password Template

Provides information that there is a new password for an account. By default, this template is enabled.

For example, the template provides this message:

Account new password information

The following is your new password for account amyaccount:

View Changes: http://localhost:9090/itim/console
Process Reference: 2855285841498421007
New password: secret
Account ID: myaccount
Account Service: ITIM Service
Account Service Profile: ITIM
Owner Name: firstname lastname
Time of service provision: Apr 25, 2007 12:54:05 IST

The subject statement is:

<RE key="password_change_subject"/>

The plain text is:

<RE><KEY><JS>if (EmailContext.getTransactionId() == '0')
{ 'newAccountPassword' } else { 'newAccountPasswordPickUp'; } </JS></KEY></RE>

<ITIMURL/>

<RE key="processRef"/>: <JS>process.id;</JS>

<JS>if (EmailContext.getTransactionId() != '0')
{ '<RE key="TRANSACTION_ID_LABEL"/>: ' + EmailContext.getTransactionId(); }
</JS>

<RE key="newPassword"/>: <JS>EmailContext.getAccountPassword();</JS>

<RE key="accountID"/>: <JS>EmailContext.getAccountUserId();</JS>

<RE key="accountService"/>:

<JS>EmailContext.getAccountServiceName();</JS>

<RE key="accountServiceProfile"/>: <RE><KEY>

<JS>EmailContext.getAccountServiceProfileName();</JS></KEY></RE>

<RE key="accountOwnerName"/>:

<JS>EmailContext.getAccountOwnerName();</JS>

<RE key="timeofprovision"/>: <RE key="readOnlyDateFormat">

<PARM><JS>(new Date()).getTime();</JS></PARM></RE>

<JS>if (EmailContext.getTransactionId() != '0')
{ '<RE key="RETRIEVE_PASSWORD_TITLE"/>

+ EmailContext.getPasswordRetrieveUrl(); } </JS>

<JS>if (EmailContext.getTransactionId() != '0')
{ '<RE key="passwordExpireLabel"/>

+ EmailContext.getPasswordExpirePeriod(); } }
</JS>

Process Completion Template

Provides information that the workflow activity has completed. By default, this template is enabled.

For example, the template provides this message:

A workflow process, 1416721862784240178, has completed.
Result Summary: Success
The following process has completed

Process Information
The subject statement is:

```
<RE key="processCompletedSubject">"<PARM><JS>process.id;</JS></PARM>"
<PARM><RE key="resultSummaryValue">"<PARM><RE><KEY>
<JS>process.STATE_PREFIX + process.resultSummary;
</JS></KEY></RE></PARM></RE>
</PARM></RE>
```

The plain text is:

```
<RE key="process_completed_message" />
```
The subject statement is:

```xml
<RE key="process_timeout_subject" />
```

The plain text is:

```xml
<RE key="process_timeout_message" />
```

**Restore Account Template**

Provides information that an account has been restored. By default, this template is enabled.

For example, the template provides this message:

Restored Account Information from IBM Tivoli Identity Manager

The following ITIM Service [ITIM] account has been restored:

View Changes:http://localhost:9090/itim/console
Process Reference:2857890686820910405
New Password:secret
Account ID:myaccount
Owner Name:firstname lastname
Time Completed:Apr 25, 2007 01:04:08 IST

The subject statement is:

```xml
<RE key="restore_account_subject"/>
```

The plain text is:

```xml
<RE key="restore_account"><PARM>
<RE key="service_name_with_profile_name"><PARM>
<JS>EmailContext.getAccountServiceName();</JS></PARM>
<PARAM><RE><KEY>
<JS>EmailContext.getAccountServiceProfileName();</JS></KEY></RE></PARAM>
</PARM>
<ITIMURL/>
<RE key="processRef"/>: <JS>process.id;</JS>
<JS>if (EmailContext.getTransactionId() != '0')
  {'<RE key="TRANSACTION_ID_LABEL"/>':
    + EmailContext.getTransactionId(); }
<RE key="newPassword"/>: <JS>EmailContext.getAccountPassword();</JS>
<RE key="accountID"/>: <JS>EmailContext.getAccountUserId();</JS>
<RE key="accountOwnerName"/>:
  <JS>EmailContext.getAccountOwnerName();</JS>
```

---

Date submitted: Aug 02, 2007 03:18:54 IST
Time Completed:
Result Summary:
Requester: org
Requestee: Person B
Subject: personb
Comment:
Detail:

The subject statement is:

```xml
<RE key="process_timeout_subject" />
```

The plain text is:

```xml
<RE key="process_timeout_message" />
```
Suspend Account Template
Provides information that an account is suspended. By default, this template is enabled.

For example, the template provides this message:
Your account has been suspended by IBM Tivoli Identity Manager
The following AD Service (RFI) [ADProfile] account has been suspended:

View Changes:http://localhost:9090/itim/console
Process Reference:2857497715286893521
Account ID:myaccount
Owner Name:firstname lastname
Time Completed:Apr 25, 2007 01:02:43 IST

The subject statement is:

The plain text is:

To-Do Reminder Template
Provides the default template for workflow reminders, which are emails that remind users about pending activities to which they not responded. By default, this template is disabled.

For example, the template provides this message:
Subject:Pending workflow action:
Case 6167063972290972180.6167064647650050990
The following request has been submitted for your approval
View Changes: http://localhost:9080/itim/console
Description: ApprovalWorkflow
Requestee: anil
Subject: anil
Request Initiated: Sep 05, 2007 05:42:18 IST
Process Reference: 6167063972290972180

Requested by process:
Process ID: 6167052766519381908
Process Name: Provision Account
Description: Provision Account Process
Requester: System Administrator
Requestee: anil
Subject: anil

This WorkItem will be escalated on: Saturday, September 8, 2007.

The subject statement is:
<originalSubject/>

The plain text is:
<textBody/>

<RE key="escalation_note"/> <escalationTime/>
Chapter 9. JavaScript extensions overview

JavaScript is used in IBM Security Identity Manager to specify identity policies, provisioning policy parameters, service selection policies, placement rules for identity feeds, and orphan account adoption.

In addition, JavaScript is used in workflows to specify transition conditions, loop conditions, JavaScript activities, activity postscripts, and workflow notification. Various scripting extensions are provided by IBM Security Identity Manager to expose useful data and services to each of these scripts. In addition to these extensions, system administrators can configure IBM Security Identity Manager to load custom JavaScript extensions. For more information about custom JavaScript extensions, see the ISIM_HOME/data/scriptframework.properties file.

IBM Security Identity Manager supports two Java Script interpreters: IBM JSEngine and Free EcmaScript Interpreter (FESI, now deprecated). Both of these interpreters support the third edition (December 1999) of the ECMA-262 specification.

Table 6 lists the supported host components and script extensions.

<table>
<thead>
<tr>
<th>Host Component</th>
<th>Supported Script Extension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccountTemplate</td>
<td>ProvisioningPolicyExtension</td>
<td>Extensions registered with this key are loaded by Account Default Template parameters.</td>
</tr>
<tr>
<td></td>
<td>ServiceExtension</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SubjectExtension</td>
<td></td>
</tr>
<tr>
<td>Delegate</td>
<td>DelegateExtension</td>
<td>Extensions registered with this key are loaded by Delegation notifications.</td>
</tr>
<tr>
<td></td>
<td>Model Extensions Package</td>
<td></td>
</tr>
<tr>
<td>HostSelection</td>
<td>ServiceModelExtension</td>
<td>Extensions registered with this key are loaded by Service Selection Policies.</td>
</tr>
<tr>
<td></td>
<td>SubjectExtension</td>
<td></td>
</tr>
<tr>
<td>IdentityPolicy</td>
<td>IdentityPolicyExtension</td>
<td>Extensions registered with this key are loaded by Identity Policies.</td>
</tr>
<tr>
<td></td>
<td>AttributesExtension</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ServiceExtension</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SubjectExtension</td>
<td></td>
</tr>
<tr>
<td>OrphanAdoption</td>
<td>Model Extensions Package</td>
<td>Extensions registered with this key are loaded by adoption scripts.</td>
</tr>
<tr>
<td></td>
<td>ServiceExtension</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SubjectExtension</td>
<td></td>
</tr>
<tr>
<td>PersonPlacementRules</td>
<td>PersonPlacementRulesExtension</td>
<td>Extensions registered with this key are loaded by placement rules during identity feeds.</td>
</tr>
<tr>
<td></td>
<td>ServiceExtension</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AttributesExtension</td>
<td></td>
</tr>
<tr>
<td>PostOffice</td>
<td>PostOfficeExtension</td>
<td>Extensions registered with this key are loaded by Post Office templates.</td>
</tr>
</tbody>
</table>
### Packaged extensions

The section describes the scripting extensions provided by IBM Security Identity Manager, the JavaScript objects they expose, and the scripts to which these extensions are applicable.

Do not remove these extensions from the properties file that you configure, because they are necessary for standard product operation. All of the extensions are configured for new installations.

#### AttributesExtension

The full extension name is `com.ibm.itim.script.extensions.AttributesExtension`.

This extension is responsible for making the `ATTRIBUTES` object available to scripts. `ATTRIBUTES` is a Map type object and is used internally to implement the deprecated `Enrole.getAttributeValue()` and `Enrole.getAttributeValues()` methods.

AttributesExtension and the `ATTRIBUTES` script object are deprecated. Do not use them in any new scripts.

**Availability**

- IdentityPolicy
- PersonPlacementRules
- ProvisioningPolicy

**JavaScript Objects**

`ATTRIBUTES`
DelegateExtension

The full extension name is com.ibm.itim.script.extensions.DelegateExtension.

This extension is responsible for making the Delegate object available to delegation notification scripts.

Availability
  Delegation Notification

JavaScript Objects
  Delegate

EmailContextExtension

The full extension name is com.ibm.itim.workflow.script.EmailContextExtension.

The EmailContextExtension provides the EmailContext object that provides information about the workflow activity and process that is making the notification. EmailContext is of type WorkflowRuntimeContext, although it might be a more specific subtype, depending on what type of change triggered the notification.

Availability
  Notification

JavaScript Objects
  EmailContext

EnroleExtension

The full extension name is com.ibm.itim.script.extensions.EnroleExtension.

This extension is automatically loaded for all scripts. It is not in the scriptframework.properties file.

This extension exposes the Enrole object to scripts. This object provides the following miscellaneous functions:
  * Converting to and from the generalized time format.
  * Logging and tracing facilities to write to the IBM Security Identity Manager logs.

Availability
  All scripts

JavaScript Objects
  Enrole
  Error

IdentityPolicyExtension

The full extension name is com.ibm.itim.policy.script.IdentityPolicyExtension.

This extension exposes the IdentityPolicy object to identity policy scripts. This object provides a method to test for the existence of a user ID.

Availability
  Identity Policy

JavaScript Objects
IdentityPolicy

LoopCountExtension
The full extension name is com.ibm.itim.workflow.script.LoopCountExtension.

This extension provides the loopcount script object. The object is an integer that tells a script how many times a loop ran.

Availability
Workflow

JavaScript Objects
loopcount

Model extensions package
The model extensions expose JavaScript objects that can be used to search for people, accounts, services, and organizational units such as organizations, business units, and locations.

Important: The objects exposed by these extensions allow access to identity and service data without regard to specified access control rules for these data. The objects are considered privileged. Define access control items that manage access to IBM Security Identity Manager scripts.

All of the model extensions have the same availability and can be used with the following extension points:
- AccountTemplate
- ProvisioningPolicy
- HostSelection
- OrphanAdoption
- Workflow
- Notification

AccountModelExtension
The full extension name is com.ibm.itim.script.extensions.model.AccountModelExtension.

This extension exposes the Account constructor and AccountSearch constructor to applicable scripts. After it is constructed, an Account object represents an Account Directory Object in scripts. The AccountSearch object provides methods to search for existing accounts based on several parameters, which include uid, owner, and service.

JavaScript Objects
- AccountSearch
- Account

CredentialModelExtension
The full extension name is com.ibm.itim.script.extensions.model.CredentialModelExtension.

This extension exposes the Credential constructor to applicable scripts when Shared Access Module is activated. When it is constructed, a Credential object represents a Credential Directory Object in scripts.
Note: You must install and enable the shared access module in order to use com.ibm.itim.script.extensions.model.CredentialModelExtension.

JavaScript Objects
  Credential

PersonModelExtension
The full extension name is com.ibm.itim.script.extensions.model.PersonModelExtension.

This extension exposes the Person constructor, PersonSearch constructor, and ExtendedPerson constructor to applicable scripts. After it is constructed, a Person object represents a Person Directory Object in script. A ExtendedPerson object extends Person with ownership type information. The PersonSearch object provides methods to search for existing people based on a provided LDAP filter.

JavaScript Objects
  • PersonSearch
  • Person
  • ExtendedPerson

OrganizationModelExtension
The full extension name is com.ibm.itim.script.extensions.model.OrganizationModelExtension.

This extension exposes the ContainerSearch constructor to applicable scripts. The ContainerSearch object provides methods to search of Organizational containers based on LDAP filters.

JavaScript Objects
  ContainerSearch

RoleModelExtension
The full extension name is com.ibm.itim.script.extensions.model.RoleModelExtension.

This extension exposes the Role constructor and RoleSearch constructor to applicable scripts. After it is constructed, the Role object represents a Role Directory Object in scripts. The RoleSearch object provides a method to search for Roles based on role name.

JavaScript Objects
  • RoleSearch
  • Role

ServiceModelExtension
The full extension name is com.ibm.itim.script.extensions.model.ServiceModelExtension.

This extension exposes the Service constructor and ServiceSearch constructor to applicable scripts. After it is constructed the Service object represents a Service Directory Object in scripts. The ServiceSearch object provides methods to search for Service based on several parameters, which include LDAP filter and service name.

JavaScript Objects
  • ServiceSearch
• Service

**PersonPlacementRulesExtension**

The full extension name is

This extension provides the entry object to the scripting environment. The entry object is of type Map and contains the attribute values for the Person that is placed.

**Availability**

PersonPlacementRules

**JavaScript Objects**

entry

**PostOfficeExtension**

The full extension name is

The Post Office capability reduces the number of email messages received by workflow participants by combining similar notifications into a single email. The emails are combined with a template specified in the system configuration pages. This extension exposes a JavaScript object, PostOffice, to JavaScript snippets in these templates. This object provides methods for accessing all the distinct emails, the email address of the recipient, the email topic, and the recipient data.

**Availability**

Post Office Template

**JavaScript Objects**

PostOffice

**ProvisioningPolicyExtension**

The full extension name is

This extension provides the scripting objects reason and parameters to the scripting environment. The reason object is an integer that informs a script of the reason the evaluation is happening: 0 if a new account or 1 if an existing account. The parameters object is a map that contains the information about the account that is being evaluated. Currently, only the uid field is supported.

**Availability**

AccountTemplate

ProvisioningPolicy

**JavaScript Objects**

parameters

reason

**ReminderExtension**

The full extension name is com.ibm.itim.script.extensions.ReminderExtension.
This extension exposes the reminderCtx object to JavaScript snippets contained in email reminders. This object provides methods for accessing the original email text and subject. It also provides the due date and time for the associated to-do item.

**Availability**
- E-mail reminders

**JavaScript Objects**
- reminderCtx

### ServiceExtension

The full extension name is `com.ibm.itim.script.extensions.ServiceExtension`.

This extension exports the service object to the scripting environment. The service object is a DirectoryObject type and represents the Service associated with a provisioning operation.

**Availability**
- IdentityPolicy
- OrphanAdoption
- PersonPlacementRules
- AccountTemplate
- ProvisioningPolicy

**JavaScript Objects**
- service

### SubjectExtension

The full extension name is `com.ibm.itim.script.extensions.SubjectExtension`.

This extension provides the subject scripting object. In all of the scripting contexts except for OrphanAdoption, subject is a DirectoryObject. In the OrphanAdoption context, subject is a Map of the attributes returned by a reconciliation.

**Availability**
- HostSelection
- IdentityPolicy
- OrphanAdoption
- Reminder
- AccountTemplate
- ProvisioningPolicy

**JavaScript Objects**
- subject

### WorkflowExtension

The full extension name is `com.ibm.itim.workflow.script.WorkflowExtension`.

This extension exposes JavaScript objects that can be used to access data from a workflow process in progress. In addition, it exposes objects that can be used to get or set the status, state, and result of a workflow process or activity.

**Availability**
- Workflows

**JavaScript Objects**
- process
• activity
• participant
• Relevant Data

**Note:** Relevant Data are objects defined by the workflow designer. Check with system administrator to find the names of specific Relevant Data objects.

**Relevant data JavaScript objects**

Each process has a set of relevant data, or process specific parameters, which can be read or changed from in a workflow script.

The name and syntax of these parameters, or relevant data items, are defined in the workflow designer and are typically specific to the workflow process purpose. For example when you add a user, an object that holds all the attributes of the new user can be a relevant data item. However, when you delete a user, the only required relevant data item can be the distinguished name of the user to delete.

Each relevant data item is represented in the workflow script as a variable with the same relevant data ID as defined in the workflow designer. These relevant data items all have the following functions:

**get()**
This function returns a JavaScript object that represents the value of the relevant data item. There is a variable present for each relevant data item in the context of the script. For performance reasons, however, the values are not retrieved from the workflow engine until the script specifically requests it with this call. The returned JavaScript object is in the same syntax as defined in the workflow designer.

Usage:
```
dn = subjectDN.get();
```

where `subjectDN` is defined as a relevant data item for the current process.

**set(Object value)**
The `set(Object value)` function changes the value of the relevant data item. It not only updates the relevant data item in the script, but also in the workflow engine. The new value is a parameter to the function. The new value must be compatible with the syntax of the relevant data item as defined in the workflow designer. For example, if the relevant data item is an integer, the value `cat` is not a valid parameter to this function.

Usage:
```
ou.set("engineering");
```

where `ou` is defined as a relevant data item for the current process.

---

**Registering JavaScript extensions**

JavaScript extensions might not be useful or applicable to every scriptable function that IBM Security Identity Manager provides. For example, an extension used by Post Office templates might not be applicable to provisioning policy parameters. An extension designed for one class of script might not load or behave appropriately when loaded into another class of script.
IBM Security Identity Manager has the classes of script that are listed in Table 7 on page 62. JavaScript extensions might be registered to load and run with any combination of these script classes.

JavaScript extensions are configured in these files:

**scriptframework.properties**

*For all new extensions. Use this file to configure script extensions and other scripting functions.*

JavaScript extensions are registered in the ISIM_HOME/data/scriptframework.properties file. This file is formatted with the standard Java Properties *key [subkey]=value* format.

Where

- **key** is the name assigned to the target script class, described in Table 7 on page 62 and the
- **value** is the full class name of the ScriptExtension interface. The
- **subkey** is used when more than one extension is registered for a script class. (Optional)

**Note:**

1. IBM Security Identity Manager is installed with a set of extensions for each script class already registered in the scriptframework.properties file. Do not remove these extensions from the file as they are necessary for the product to function correctly.
2. To prevent the possibility of a code injection attack, do not use the JavaScript function `eval()`.
3. By default, only the set of extensions registered in the scriptframework.properties file is available for the particular script. You can configure any supported extension for the script by registering JavaScript extensions in the scriptframework.properties file. For information about supported script extensions, see Table 6 on page 53. For information about the properties and methods available for each JavaScript extension object, see Chapter 10, “JavaScript extension reference,” on page 69.

**fesiextensions.properties**

*Provides support for Free ECMAScript Interpreter (FESI) JavaScript extensions before Version 5.0 of IBM Tivoli Identity Manager. Do not author new extensions with this deprecated architecture.*

If you continue to use the deprecated fesiextensions.properties file, save the fesi.jar library in the ITIM_HOME/lib directory before you upgrade IBM Security Identity Manager to Version 5.0. Replace the newly installed file with the custom fesi.jar file after the upgrade completes.

The following line registers a single extension for use in IBM Security Identity Manager scripts:

```
```

These example lines register multiple extensions for use in IBM Security Identity Manager scripts:

```
ITIM.extension.IdentityPolicy.1=com.ibm.itim.policy.script.IdentityPolicyExtension
ITIM.extension.IdentityPolicy.2=com.yourcompany.script.YourCustomExtension
```
### Configuring scriptframework.properties

Use the `ISIM_HOME/data/scriptframework.properties` file, which provides extended documentation for these tasks, to configure major scripting functions.

The major scripting functions include:

**Extensions**

Specifies which extensions to load for each host component. To load more than a single extension for any host component, add a suffix to the properties key (each key must be unique). For example:

```
ITIM.extension.IdentityPolicy.service=com.ibm.itim.script.extensions.ServiceExtension
```

**Interpreters**

Configures the interpreter to use for each host component. The default is the IBM JSEngine. The other option is FESI, which can be used only if the fesi.jar file exists in `ISIM_HOME/lib/`. It is only to be used by customers of IBM Tivoli Identity Manager Version 4.6 and earlier who wrote their own custom FESI extensions.

**Wrappers**

All objects available to scripts are really Java objects used by IBM Security Identity Manager. To prevent security issues, IBM Security Identity Manager wraps these objects in wrappers. Use this section of the `scriptframework.properties` file to change the default wrappers used by IBM Security Identity Manager. Default scripts provided by IBM Security Identity Manager assume the use of default wrappers. If you change the scripts, functions might stop working. This section is for advanced use only.

**Miscellaneous**

Determines whether profiling information is collected and included in the trace log and whether plain text passwords can be accessed from Person and Account objects.
Migration of custom FESI extensions to the IBM JSEngine

Migration of a custom FESI extension to a script extension makes your code shorter, easier to read, and easier to understand.

Note: Support for FESI is deprecated in IBM Security Identity Manager Version 6.0.

For detailed information and examples about how to write new extensions, see the documentation in ISIM_HOME/extensions/doc/javascript/javascript.html.

The following example illustrates the migration steps.

**Best practice in handling function returns**

You can minimize problems that might occur due to differences in how FESI and IBM JSEngine handle JavaScript. The differences involve implicit return values from functions.

For example, given these statements:

```javascript
function sumValue() {
  var a = 3;
  var b = 2;
  a + b;
}
```

With FESI, the function `sumValue()` returns 5 because 5 is the result of the last statement run in the function. Using IBM JSEngine, the expression `sumValue()` returns null because there is no explicit return. The correct code for IBM JSEngine includes an explicit return statement:

```javascript
function sumValue() {
  var a = 3;
  var b = 2;
  return a + b;
}
```

To keep JavaScript code consistent, always use an explicit return value in functions. In the previous release, some of the service selection script examples did not use an explicit return value. Update any JavaScript code that is based on these examples to have an explicit return value, to ensure that the code continues to work after an upgrade to use IBM JSEngine.

**Plain Old Java Object (POJO) example**

Start with a Plain Old Java Object (POJO, in this example) that contains all of the business logic for your extension.

For example:

```java
public class Extension {
    public static void log(String msg) {
        System.out.println(msg);
    }
}
```

In this case, the POJO contains a single method. Your typical extension contains more logic. For example:

```java
static class FESIExtension implements JSExtension {
    public void initializeExtension(JSGlobalObject go) throws JSException {
        // Create the prototype
```
final JSObject prototype = go.makeJSObject();

prototype.setMember("log", new JSFunctionAdapter() {
    public Object doCall(JSObject thisObject, Object[] args)
        throws JSException {
        if (args.length >= 1) {
            Extension.log(args[0].toString());
        }
        return null;
    }
});

final JSObject obj = go.makeJSObject(prototype);

// This is the name of the object to be used in JavaScript Code
go.setMember("CustomExtension", obj);

go.setMember("log", new JSFunctionAdapter() {
    public Object doCall(JSObject thisObject, Object[] args)
        throws JSException {
        if (args.length >= 1) {
            Extension.log(args[0].toString());
        }
        return null;
    }
});

go.setMember("Logger", new JSFunctionAdapter() {
    public Object doNew(JSObject thisObject, Object[] args)
        throws JSException {
        JSGlobalObject go = thisObject.getGlobalObject();
        JSObject proto = go.makeJSObject();

        proto.setMember("log", new JSFunctionAdapter() {
            public Object doCall(JSObject thisObject, Object[] args)
                throws JSException {
            if (args.length >= 1) {
                Extension.log(args[0].toString());
            }
            return null;
        });

        final JSObject obj = go.makeJSObject(proto);
        return obj;
    }
});

This FESI extension has three main parts:

1. First, the extension makes a JSObject named prototype and adds the method "log" to prototype:
   
   final JSObject prototype = go.makeJSObject();
   
   prototype.setMember("log", new JSFunctionAdapter() {
       public Object doCall(JSObject thisObject, Object[] args)
           throws JSException {
           if (args.length >= 1) {
               Extension.log(args[0].toString());
           }
           return null;
       }
   });
The prototype `JSObject` is then added to the `JSGlobalObject` with the name `CustomExtension`. This addition allows scripts to call:
```
CustomExtension.log("message");
```

2. The second part of the extension creates a global function named `log`.
```
go.setMember("log", new JSFunctionAdapter() {
    public Object doCall(JSObject thisObject, Object[] args)
        throws JSException {
        if (args.length >= 1) {
            Extension.log(args[0].toString());
        }
        return null;
    }
});
```

Now, a script can call:
```
log("message");
```

3. The third part of the extension creates a constructor that can be called from scripts. For example:
```
go.setMember("Logger", new JSFunctionAdapter() {
    public Object doNew(JSObject thisObject, Object[] args)
        throws JSException {
        JSGlobalObject go = thisObject.getGlobalObject();
        JSObject proto = go.makeJSObject();
        proto.setMember("log", new JSFunctionAdapter() {
            public Object doCall(JSObject thisObject, Object[] args)
                throws JSException {
            if (args.length >= 1) {
                Extension.log(args[0].toString());
            }
            return null;
        });
        final JSObject obj = go.makeJSObject(proto);
        return obj;
    }
});
```

With this constructor, scripts can do the following:
```
var logger = new Logger();
logger.log("message");
```

**Conversion to a script extension**

When you convert a FESI extension to a script extension, the root of a script extension is the `ScriptExtension` interface.

You must implement this interface to create script extension.
```
public class ITIMEExtension implements ScriptExtension {
    public List getContextMenu() {
    }

    public void initialize(ScriptInterface si, ScriptContextDAO dao)
        throws ScriptException, IllegalArgumentException {
    }
}
```

Chapter 9. JavaScript extensions overview  65
To create object that can be used in scripts, create a POJO class that contains all of the business logic, and implements the marker interface ExtensionBean. A marker interface means that ExtensionBean does not require you to implement any methods and it does add any new data to your class. A POJO that implements ExtensionBean is treated specially by the IBM Security Identity Manager scripting components.

If your class does not implement ExtensionBean, then scripts cannot use the methods provided by the POJO. For example:

```java
public class Extension implements ExtensionBean {
    public static void log(String msg) {
        System.out.println(msg);
    }
}
```

In the initialize method of your extension, create ContextItem that contains an instance of your extension and add that ContextItem to a List.

```java
ContextItem custom = ContextItem.createItem("CustomExtension",
    new Extension());
items.add(custom);
```

To create global function, use ContextItem, but this time call createGlobalFunction. For example:

```java
ContextItem func = ContextItem.createGlobalFunction("log",
    new GlobalFunction() {
        public Object call(Object[] parameters)
            throws ScriptEvaluationException {
            if (parameters.length >= 1) {
                Extension.log(parameters[0].toString());
            }
            return null;
        }
    });
items.add(func);
```

The second argument to createGlobalFunction is a GlobalFunction object. GlobalFunction has a single method that you must implement and call. It is similar to the doCall method from the FESI JSFunctionAdapter. GlobalFunctions are not suggested because, like the doCall method, they pass an array of parameters. You must check that all of the parameters exist and are the right types, which can be difficult to maintain over the life of your extension.

### Creation of a constructor

To create a constructor, use ContextItem and the createConstructor method.

For example:

```java
ContextItem logger = ContextItem.createConstructor("Logger",
    Extension.class);
items.add(logger);
```

The second parameter to createConstructor is the Class object for the object that you want to construct. It is usually a POJO that implements ExtensionBean.

In each of these examples, you add the ContextItem to a List. In the getContextItems method of ScriptExtension, you return that List. For example, the full code is:
public class ITIMExtension implements ScriptExtension {
    private List<ContextItem> items;
    
    public List getContextItems() {
        return items;
    }
    
    public void initialize(ScriptInterface si, ScriptContextDAO dao)
            throws ScriptException, IllegalArgumentException {
        items = new ArrayList<ContextItem>();
        ContextItem custom = ContextItem.createItem("CustomExtension",
                new Extension());
        items.add(custom);
        ContextItem func = ContextItem.createGlobalFunction("log",
                        new GlobalFunction() {
                            public Object call(Object[] parameters)
                                    throws ScriptEvaluationException {
                                if (parameters.length >= 1) {
                                    Extension.log(parameters[0].toString());
                                }
                                return null;
                            }
                        });
        items.add(func);
        ContextItem logger = ContextItem.createConstructor("Logger",
                        Extension.class);
        items.add(logger);
    }
}

Download of fesi.jar from a non-IBM source (deprecated)

If you want to use FESI, but do not have the required libraries, you can download
and enable the libraries.

Note: Support for FESI is deprecated in IBM Security Identity Manager Version
6.0.

If you upgrade from IBM Tivoli Identity Manager Version 4.6, do not follow these
steps. The correct version of the FESI library is maintained during the upgrade.
1. Download FESI version 1.1.8 from http://www.lugin.ch/fesi/. At the time of
   this writing, 1.1.8 is the latest version
   a. Follow the link to download the current version, which displays the license
      page.
   b. Accept the license to continue.
   c. Access a download page.
   d. Download the install-fesi-1.1.8.jar file.
2. After the file downloads successfully, start the installer by typing this command
   from the command line:
   java –jar install-fesi-1.1.8.jar
   a. Follow the remaining steps that the installer provides to install FESI.

Note: Because IBM Security Identity Manager needs only a JAR file from
the installation, you can install FESI to a temporary location that you can
delete later.
b. After the installation completes, go to $FESI_INSTALL_DIR/lib and locate the fesi.jar file.
c. Copy the fesi.jar file to the ISIM_HOME/lib directory.

   a. Log on to the WebSphere® Application Server Administrative Console, which is typically at http://hostname:9060/ibm/console, where hostname is specific to your computer.
   b. Go to Environment > Shared Libraries > ITIM_LIB.
   c. At the bottom of the Classpath text box, add the line ${ISIM_HOME}/lib/fesi.jar.

4. Restart the WebSphere Application Server to put the changes into effect.

5. Edit the scriptframework.properties file to use the FESI interpreter. When you are using FESI, the script framework looks for the fesiextensions.properties file to determine which FESI extensions to load. If this file does not exist, a message is written to the trace.log file for every script that is run by FESI.
Chapter 10. JavaScript extension reference

The reference section is arranged alphabetically.

There are a number of IBM Security Identity Manager specific objects available for use. IBM Security Identity Manager uses JavaScript extensions to package JavaScript objects and APIs. An extension can also be a package of other extensions (for example, ModelExtension). After an extension is defined, it can be registered in the ISIM_HOME/data/scriptframework.properties file to be used in a specific JavaScript context. In some cases, an environment needs to be created for an extension.

Table 8 shows these script extensions.

<table>
<thead>
<tr>
<th>Script Extension</th>
<th>Object Name</th>
<th>Object Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttributesExtension (deprecated)</td>
<td>ATTRIBUTES</td>
<td>Map</td>
</tr>
<tr>
<td>EmailContextExtension</td>
<td>EmailContext</td>
<td>EmailContext</td>
</tr>
<tr>
<td>EnroleExtension</td>
<td>Enrole</td>
<td>Enrole Error</td>
</tr>
<tr>
<td>IdentityPolicyExtension</td>
<td>IdentityPolicy</td>
<td>IdentityPolicy</td>
</tr>
<tr>
<td>LoopCountExtension</td>
<td>loopcount</td>
<td>int</td>
</tr>
<tr>
<td>PersonPlacementRulesExtension</td>
<td>entry</td>
<td>Map</td>
</tr>
<tr>
<td>PostOfficeExtension</td>
<td>PostOffice</td>
<td>PostOffice</td>
</tr>
<tr>
<td>ProvisioningPolicyExtension</td>
<td>parameters</td>
<td>Map</td>
</tr>
<tr>
<td></td>
<td>reason</td>
<td>int (0: New Account, 1: Existing Account)</td>
</tr>
<tr>
<td>AccountModelExtension</td>
<td>Account constructor</td>
<td>Account</td>
</tr>
<tr>
<td></td>
<td>AccountSearch constructor</td>
<td>AccountSearch</td>
</tr>
<tr>
<td>CredentialModelExtension</td>
<td>Credential</td>
<td>Credential</td>
</tr>
<tr>
<td>OrganizationModelExtension</td>
<td>ContainerSearch constructor</td>
<td>ContainerSearch</td>
</tr>
<tr>
<td>PersonModelExtension</td>
<td>Person constructor</td>
<td>Person</td>
</tr>
<tr>
<td></td>
<td>ExtendedPerson constructor</td>
<td>ExtendedPerson</td>
</tr>
<tr>
<td></td>
<td>PersonSearch constructor</td>
<td>PersonSearch</td>
</tr>
<tr>
<td>RoleModelExtension</td>
<td>Role constructor</td>
<td>Role</td>
</tr>
<tr>
<td></td>
<td>RoleSearch constructor</td>
<td>RoleSearch</td>
</tr>
<tr>
<td>ServiceModelExtension</td>
<td>Service constructor</td>
<td>Service</td>
</tr>
<tr>
<td></td>
<td>ServiceSearch</td>
<td>ServiceSearch</td>
</tr>
<tr>
<td>ReminderExtension</td>
<td>reminderCtx</td>
<td>Reminder</td>
</tr>
<tr>
<td>ServiceExtension</td>
<td>service</td>
<td>DirectoryObject</td>
</tr>
</tbody>
</table>
Table 8. Script extensions (continued)

<table>
<thead>
<tr>
<th>Script Extension</th>
<th>Object Name</th>
<th>Object Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubjectExtension</td>
<td>subject</td>
<td>Person</td>
</tr>
<tr>
<td>Note: For Orphan Adoption Rule JavaScript, the subject is a Map, which contains the account attributes returned from reconciliation. The entries in the map are referred by the name of the account attributes, which might vary based on the service type.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WorkflowExtension</td>
<td>process activity</td>
<td>Activity</td>
</tr>
<tr>
<td>Participant constructor</td>
<td>ParticipantType</td>
<td>ProcessData Process</td>
</tr>
</tbody>
</table>

Finding methods and properties for a specific JavaScript object

This example demonstrates how to find methods and properties for a specific JavaScript object.

If you are writing a workflow script, look in the scriptframework.properties file to see which extensions are available. By default, workflow loads the model extensions, the WorkflowExtension, and the LoopCountExtension.

Table 8 on page 69 shows that WorkflowExtension defines scripting objects that include process, activity, a Participant constructor, an object named ParticipantType, and a series of workflow-specific pieces of data.

In another column in the table, notice that the process object is of type Process. Now, locate Process in this reference to see that Process type has a property called name, and a method called getParent().

To understand how to use maps, notice that objects, such as parameters from ProvisioningPolicyExtension, have a type of Map. A Map, also known as a dictionary, is a named JavaScript object that can hold many other objects which can be accessed by name. The parameters object holds another object named uid. To access uid, you can type parameters.uid[0]. (In this case uid is an array, so you must type [0] to get the first element of the array.) The values that a map holds will vary between each map. For more information, locate the specific map in the JavaScript reference.

How to read the reference pages

This section explains the structure of each reference item.

**Title and Description**

Every reference entry begins with a title and a one line description. The entries are alphabetized by title. The one-line description gives a quick summary of the item documented in the entry.

**Availability**

The IBM Security Identity Manager JavaScript extensions change over time. Unless otherwise noted, anything available in one version of the IBM Security Identity Manager extensions is also available in later versions. This section also specifies whether an existing item was enhanced with a
later version of the extensions and when an item is deprecated. Deprecated items are no longer supported and can be removed from future versions of the IBM Security Identity Manager extensions. Do not use deprecated items in new IBM Security Identity Manager JavaScript code.

Provided by
At installation, IBM Security Identity Manager provides this initial set of registered extensions:

- EnroleExtension
- ProvisioningPolicyExtension
- PostOfficeExtension
- IdentityPolicyExtension
- PersonPlacementRulesExtension
- WorkflowExtension
- ReminderExtension
- ServiceExtension
- SubjectExtension
- AttributesExtension
- LoopCountExtension
- EmailContextExtension
- Model extensions package

Inherits From
JavaScript classes can inherit properties and methods from other classes. When it occurs, an Inherits From section appears in the reference entry. The inherited fields and methods are in the listed superclasses. For example, the subject object inherits all of its fields and properties from the DirectoryObject class.

Synopsis
This section is a synopsis of how to use the object, method, property, or function.

Arguments
If the reference page describes a function or method that has arguments, the Synopsis is followed by an Arguments subsection that describes the arguments to the function or method. For some objects, the Synopsis section is replaced by a Constructor section which is also followed by an Arguments subsection.

Returns
If a function or a method has a return value, the Arguments subsection is followed by a Returns subsection that explains the return value of the function, method or constructor.

Properties
If the reference page documents an object, the Properties section lists the properties the object supports and provides short explanations of each.

Methods
The reference page for an object that defines methods includes a Methods section.

Description
Most reference entities contain a Description section, which is a basic description of whatever is documented. For some simple methods, the
Arguments and Returns sections document the method sufficiently by themselves, so the Description section is omitted.

Usage  This section describes common techniques for using the item, or it contains cautionary information.

---

**Account**

Represents an account that is associated with a provisioning operation.

**Availability**

IBM Tivoli Identity Manager 4.x.

**Inherits From**

DirectoryObject

**Provided by**

com.ibm.itim.script.extensions.model.AccountModelExtension

**Constructor**

```java
new Account(dn)
```

**Returns**

The newly created Account object that represents the account with the specified DN, which is a String.

---

**AccountSearch**

You can search for an account with the AccountSearch object.

**Availability**

IBM Tivoli Identity Manager 4.x.

**Provided by**

com.ibm.itim.script.extensions.model.AccountModelExtension

**Constructor**

```java
new AccountSearch()
```

**Returns**

The newly created and initialized account search object.

**Methods**

- `searchByOwner()`  
  Search for an account by owner.

- `searchByUid()`  
  Search for an account by user ID.

- `searchByUidAndService()`  
  Search for an account by user ID and service.

- `searchByURI()`  
  Search for an account by URI within an organizational container.

**Description**

The entity implements the IBM Security Identity Manager Account Search class.

---

**AccountSearch.searchByOwner()**

The method finds an account entity by the distinguished name of the owner.
Availability
IBM Tivoli Identity Manager 4.x.

Synopsis

AccountSearch.searchByOwner(personDN)

Arguments

personDN
String representing the distinguished name of the account owner.

Description
Given the distinguished name of the person, find the account entities owned by that person. This function will return null if the person is not found.

Usage

```javascript
var account = (new AccountSearch()).searchByOwner(person.dn);
if (account!=null) {
    Enrole.log("script", "Found " + account.length + " accounts");
}
```

AccountSearch.searchByUid()

The method finds an account entity by user ID and distinguished name of a service.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis

AccountSearch.searchByUid(uid, serviceDN)

Arguments

uid
String representing the user ID of the account.

serviceDN
String representing the distinguished name of the account.

Description
Given the user ID of the account and the distinguished name of the service, find the account entity. This function returns null if there is not exactly one matching account, or if the service is not found.

Usage

```javascript
var account = (new AccountSearch()).searchByUid("pallen", service.dn);
if (account!=null) {
    Enrole.log("script", "Found account pallen");
}
```

AccountSearch.searchByUidAndService()

The method finds an account entity by user ID, service name, and service profile name.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis

AccountSearch.searchByUidAndService(uid, serviceName)

Arguments

uid
String representing the user ID of the account.
serviceName

String representing the name of the service.

Description
Given the user ID of the account and the name of the service that has the same service profile as the script context service profile, find the account entity. This function returns null if:

• More than one matching account exists.
• The service is not found.
• More than one service with the given name exists.

Usage
```
var account = (new AccountSearch()).searchByUidAndService
  ("pallen", "NT Domain Controller");
if (account!null) {
  Enrole.log("script", "Found account pallen");
}
```

Synopsis
AccountSearch.searchByUidAndService(uid, serviceName,
  serviceProfileName)

Arguments
uid    String representing the user ID of the account.
serviceName    String representing the name of the service.

Usage
```
var account = (new AccountSearch()).searchByUidAndService
  ("pallen", "NT Domain Controller", "ADProfile");
if (account!null) {
  Enrole.log("script", "Found account pallen");
}
```

AccountSearch.searchByURI()

The method finds an account by URI in an organizational container.

Availability
IBM Security Identity Manager 6.0

Synopsis
AccountSearch.searchByURI(containerDN, uri)

Arguments
Container DN    String representing the distinguished name of the organizational container.
uri    String representing the URI of the account.
Returns
An Account object.

Description
Given the distinguished name of an organizational container and the
account URI, this method finds the account. If the account is not found,
this function returns null. If more than one account is found, this function
throws a scripting exception.

Usage
var account = (new AccountSearch()).searchByURI(container.dn, uri);
if (account != null)
    Enrole.log("script", "Found " + account.getProperty("eruid") );

Activity
Activity is used to reference any activity in a IBM Security Identity Manager
workflow.

Availability
IBM Tivoli Identity Manager 4.x

Provided by
The activity JavaScript object in the WorkflowExtension returns an Activity
object that represents the current workflow activity. The workflow activity
can be used in the context of a workflow activity PostScript, or in a
transition script, to reference the current activity. For a transition script, this
object represents the activity whose completion has lead to the evaluation
of the transition script.

Process.getActivity() can return any Activity object in the context of a
workflow process. For more information, see the description of this
method.

Activity Result Summary Code

APPROVED
Approved process summary code. Result code is AA.

ESCALATED
Escalated process summary code. Result code is ES.

FAILED
Failed process summary code. Result code is SF.

PARTICIPANT_RESOLVE_FAILED
Participant resolved failure process summary code. Result code is PF.

PENDING
Pending process summary code. Result code is PE.

REJECTED
Rejected process summary code. Result code is AR.

SUBMITTED
Submitted process summary code. Result code is RS.

SUCCESS
Success process summary code. Result code is SS.

TIMEOUT
Time out process summary code. Result code is ST.
WARNING
Warning process summary code. Result code is Sw.

Properties

description
Describes the purpose of the activity given when defined in the workflow designer.

duedate
Indicates the time in milliseconds by when the activity is due.

id
Assigned by the workflow designer to uniquely identify the workflow activity within the workflow engine.

index
Index of the instance of the activity.

name
Label given this activity when defined in the workflow designer.

participant
The activity participant, as defined in the workflow designer.

resultDetail
An application-specific string that provides more detail about the result of the activity.

resultSummary
An application-specific string that represents the summary result of the activity.

started
Indicates when the activity started.

state
Code that represents the current state of the activity.

subtype
Code that further categorizes the activity beyond the type of the activity, such as approval or request for information.

type
Code that categorizes the activity given when defined in the workflow designer, such as manual or application.

Methods

auditEvent()
Create an event in the audit trail specific to the activity.

setResult()
Change the result member of the activity in the current activity.

Description
This entity represents the current workflow activity that is being run. Within the context of a workflow transition script, this entity represents the activity whose completion has led to the evaluation of the transition script. No constructor is available to create this object in any IBM Security Identity Manager context.

Activity.auditEvent()
The method creates an event in the audit trail.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis

activity.auditEvent(event)
Arguments

   event  String representing the event to be audited.

Description
This method creates an event in the audit trail specific to the activity. The function takes in one parameter that can be any JavaScript object that can be translated into a String for storage. In the audit trail, the event is automatically time stamped.

Usage  activity.auditEvent("Task completed");

Activity.description
The field provides information about the purpose of the activity.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis
activity.description

Description
This read-only field is a String that describes the purpose of the activity given when defined in the workflow designer.

Usage  x = activity.description;

Activity.duedate
The field represents the time in milliseconds by when the activity is due.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis
activity.duedate

Description
This read-only field is a long number of milliseconds by when this activity is due.

Usage  x = activity.duedate;

Activity.guid
The generated unique identifier assigned to the activity at runtime.

Availability
IBM Tivoli Identity Manager 5.x

Synopsis
activity.guid

Description
This read-only field is a String of the generated unique identifier for the workflow activity within the workflow engine.

Usage  x = activity.guid;

Activity.id
The field is the unique identifier assigned to the activity.
Availability
IBM Tivoli Identity Manager 4.x.

Synopsis
\texttt{activity.id}

Description
This read-only field is a String assigned by the workflow designer to uniquely identify the workflow activity within the workflow engine.

Usage \hspace{1cm} \texttt{x = activity.id;}

\textbf{Activity.index}

The field is an index of the instance of the activity.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis
\texttt{activity.index}

Description
This field is a read-only and a number. If there is more than one instance of this activity, such as in the case where the activity of the ID is called multiple times in a loop in the workflow process, the value starts at one. If there is only one instance of this activity, the index value is zero.

Usage \hspace{1cm} \texttt{x = activity.index;}

\textbf{Activity.name}

The field is the label that is assigned to the activity.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis
\texttt{activity.name}

Description
This read-only field is a String assigned by the workflow designer to label this activity.

Usage \hspace{1cm} \texttt{x = activity.name;}

\textbf{Activity.participant}

The field represents the activity participant.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis
\texttt{activity.participant}

Description
This read-only field is a Participant that represents the activity participant. Not all activities have a participant. If there is no participant associated with the activity, this member is empty.

Usage \hspace{1cm} \texttt{x = activity.participant;}

78 IBM Security Identity Manager Version 6.0: Reference Guide
Activity.resultDetail
You can get the details about the result of the activity with this field.
Availability
IBM Tivoli Identity Manager 4.x.
Synopsis
activity.resultDetail
Description
This read-only field is an application-specific string that provides more
detail about the result of the activity.
Usage
x = activity.resultDetail;

Activity.resultSummary
The field helps you view the summary of the result of the activity.
Availability
IBM Tivoli Identity Manager 4.x.
Synopsis
activity.resultSummary
Summary
This read-only field is an application-specific string that provides a
summary of the result of the activity. It can represent a success or failure.
Usage
x = activity.resultSummary;

Activity.setResult()
The method changes the result member of the activity.
Availability
IBM Tivoli Identity Manager 4.x.
Synopsis
activity.setResult(summary)
activity.setResult(summary, detail)
Arguments
summary
String code that represents the result summary.
detail
String representing the result details.
Description
This method changes the result member of the activity in the current
activity. It is supported for current activities in the current workflow
process. The result is composed by an application-specific summary code,
and optional more detailed application-specific description. The summary
code can indicate a success or failure. This summary code is stored as the
resultSummary member locally and updated in the relevant data in the
workflow engine. The detail is stored as the resultDetail member locally
and updated in the relevant data in the workflow engine.
Usage
activity.setResult(activity.FAILED);
activity.setResult(activity.FAILED, "Unable to connect to resource");
**Activity.started**

The field represents the date that indicates when the activity started.

*Availability*

IBM Tivoli Identity Manager 4.x.

*Synopsis*

`activity.started`

*Description*

This read-only field is a string that represents the date that indicates when the activity started.

*Usage*

```java
x = activity.started;
```

**Activity.state**

The field represents the current state of the activity.

*Availability*

IBM Tivoli Identity Manager 4.x

*Synopsis*

`activity.state`

*Description*

This read-only field is a code string that represents the current state of the activity. The state can have the following values:

- **R** for running
- **I** for not started
- **T** for terminated
- **A** for aborted
- **S** for suspended
- **C** for completed
- **B** for bypassed

*Usage*

```java
if (activity.state == "S") {
    ...
}
```

**Activity.subtype**

The field represents the subtype of the activity.

*Availability*

IBM Tivoli Identity Manager 4.x.

*Synopsis*

`activity.subtype`

*Description*

This read-only field is a code string that further categorizes the activity beyond the type of the activity, such as approval or request for information. This is defined in the workflow designer. Not all activities have a subtype. If there is no subtype associated with the activity, this member is empty. The currently supported subtypes are:

- **AP** for approval
- **RI** for request for input
Activity.type
The field represents the type of the activity.
Availability
IBM Tivoli Identity Manager 4.x.
Synopsis
activity.type
Description
This read-only field is code string that categorizes the activity given when defined in the workflow designer, such as manual or application. The currently supported types are:
• S for subprocess
• L for loop
• A for application
• R for route
• M for manual
• O for operation
Usage x = activity.type;

AttributeChangeOperation
The object represents an entity about the attribute change operation.
Availability
IBM Tivoli Identity Manager 4.x.
Provided by
AttributeChangeOperation objects are returned from the method DirectoryObject.getChanges() and are therefore not provided by any specific extension.
Properties
attr Name of the attribute that is being changed.
op An integer that identifies the type of change that is being made.
values[] An array of objects that must be either added, removed, or replaced.
Description
This entity represents the changes made to a IBM Security Identity Manager object.

AttributeChangeOperation.attr
Represents the name of an attribute that is being changed.
Availability
IBM Tivoli Identity Manager 4.x.
Synopsis
attributeChangeOperation.attr
**Description**
Value is the attribute that is being changed.

**Usage**
x = attributeChangeOperation.attr;

**AttributeChangeOperation.op**
The field represents the type of change that is being made.

**Availability**
IBM Tivoli Identity Manager 4.x.

**Synopsis**
attributeChangeOperation.op

**Description**
This read-only field is a number that identifies the type of change that is being made. The values are:
- 1 for add
- 2 for replace
- 3 for remove

**Usage**
x = attributeChangeOperation.op;

**AttributeChangeOperation.values[]**
The field represents the name of attribute that is being changed.

**Availability**
IBM Tivoli Identity Manager 4.x.

**Synopsis**
attributeChangeOperation.values[]

**Description**
This read-only field is an array of objects that must be added, removed, or replaced.

**Usage**
x = attributeChangeOperation.values[1];

---

**ContainerSearch**
The object represents the search for an organizational container.

**Availability**
IBM Tivoli Identity Manager 4.x.

**Provided by**
com.ibm.itim.script.extensions.model.OrganizationModelExtension

**Constructor**
new ContainerSearch()

**Returns**
The newly created and initialized container search object.

**Methods**
searchByFilter()
Search for a container with a filter.
searchByURI()
Search for an organizational container by URI within a parent organizational container.

Description
Implements the IBM Security Identity Manager OrganizationalContainerSearch class.

ContainerSearch.searchByFilter()
The method represents the search for a container with a filter.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis
containerSearch.searchByFilter(profileName, filter, scope)

Arguments

profileName
The String name of the organizational container profile to use.

filter
LDAP search filter String that defines the criteria for returned containers to meet. The filter must be in the format defined by RFC2254.

scope
Optional Int search scope. Use 1 for One Level Scope and 2 for SubTree Scope. One Level Scope is the default scope.

Returns
An array of DirectoryObjects representing the results of the search.

Description
This method searches for a container with a filter.

Usage
var locationContainer = new ContainerSearch();
// use subtree scope
var thisLocation = locationContainer.searchByFilter("Location", 
"(l=Raleigh)", 2);

// use default one level scope
var otherLocation = locationContainer.searchByFilter("Location", 
"(l=Raleigh)");

ContainerSearch.searchByURI()
The method finds an organizational container by URI in a parent organizational container.

Availability
IBM Security Identity Manager 6.0

Synopsis
ContainerSearch.searchByURI(containerDN, uri)

Arguments

Container DN
String representing the distinguished name of the parent organizational container.

uri
String representing the URI of the organizational container.
Returns
A DirectoryObject representing the container.

Description
Given the distinguished name of the parent organizational container and
the container URI, this method finds the container. If the container is not
found, this function returns null. If more than one container is found, this
function throws a scripting exception.

Usage
var container = (new ContainerSearch()).searchByURI(parentContainer.dn,
    uri);
if (container != null) {
    Enrole.log("script", "Found " + container.getProperty("ou") );

Context
The object represents the context of the currently running workflow process (for
example, requestor or subject). Only used for entitlement workflows.

Note: This object type is deprecated. Use workflow JavaScript objects, such as
Process, Activity, and Relevant Data.

Some account-specific functions of the context JavaScript extension, including
getService(), isAccountDataChanged(), and getAccountParameter() cannot be
applicable to operation workflows that are not account related. The context
JavaScript extension is not suggested for custom workflows.

Availability
IBM Tivoli Identity Manager 4.x.

Provided by
com.ibm.itim.workflow.script.WorkflowExtension

Context Constants

APPROVED
This constant is used to describe the result of an activity. The
member applies only to Approval types of activities.

Usage
if (context.getActivityResult() == context(APPROVED) {...

REJECTED
This constant is used to describe the result of an activity. This
member applies only to Approval types of activities.

Usage
if (context.getActivityResult() == context.REJECTED) {...

NEWACCOUNT
This constant is used to identify the type of request that triggers
the custom workflow run time.

Usage
if (context.getProcessType() ==
    context.NEWACCOUNT) {...

ACCOUNTDATACHANGE
This constant is used to identify the type of request that triggers
the custom workflow in run time.
Usage
   if (context.getProcessType() ==
      context.ACCOUNTDATACHANGE) {...

Methods

gGetAccountParameter()
   Returns the value of an account attribute.

gActivityResult()
   Returns the activity result for the current activity.

gActivityResultByID()
   Returns the activity result for a specific activity.

gGetLoopCount()
   Returns the loop count for the current loop activity.

gGetLoopCountByID()
   Returns the current loop count for a specific loop activity.

gGetProcessType()
   Returns the type of the request that triggers the custom workflow
   process.

gGetRequestee()
   Returns the requestee associated with the request as a Person
   object.

gGetService()
   Returns the target service as a Service entity object.

isAccountDataChanged()
   Identifies whether a specific account attribute was changed in
   the request that triggers the custom workflow process.

Description
The context of the currently running workflow process (for example,
requestor or subject) is represented within the JavaScript as an object
named context.

Context.getAccountParameter()

The method returns the value of an account attribute.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis
   context.getAccountParameter(String attributeName)

Arguments
   attributeName
      String representing the attribute name.

Returns
   String value of an account attribute.

Description
   This member function returns the value of an account attribute as a string.

Usage
   parameter=context.getAccountParameter("group");
**Context.getActivityResult()**

The method returns the activity result for the current activity.

**Availability**
IBM Tivoli Identity Manager 4.x.

**Synopsis**
context.getActivityResult()

**Returns**
String

**Description**
This member function returns the activity result for the current activity. The function returns APPROVED or REJECTED. If this function is used to specify a transition condition, the function refers to the activity from which the transition is coming.

**Usage**
if (context.getActivityResult() == context.APPROVED) {...

**Context.getActivityResultById()**

The method returns the activity result for a specific activity.

**Availability**
IBM Tivoli Identity Manager 4.x.

**Synopsis**
context.getActivityResultById(String activityDefinitionID)

**Arguments**

- activityDefinitionID
  String ID of the activity definition.

**Returns**
String

**Description**
This member function returns the activity result for a specific activity. The function returns APPROVED or REJECTED.

**Usage**
if (context.getActivityResultById("1234567890") == context.APPROVED) {...

**Context.getLoopCount()**

The method returns the loop count for the current loop activity.

**Availability**
IBM Tivoli Identity Manager 4.x.

**Synopsis**
getLoopCount()

**Returns**
Integer of loop count.

**Description**
This member function returns the loop count for the current loop activity. If this function is called before a loop is started, the loop count is 0. If this activity is called while the loop activity is in process, the loop count is the
number of times the loop ran. If this function is called after the loop is completed, the loop count is the total number of times the loop is defined to run.

Usage currentIteration = context.getLoopCount();

Context.getLoopCountByID()

The method returns the current loop count for a specific loop activity.

Availability IBM Tivoli Identity Manager 4.x.

Synopsis context.getLoopCountByID(String activityDefinitionID)

Arguments

activityDefinitionID

ID of the activity definition.

Returns

Integer

Description

This member function returns the current loop count for a specific loop activity. If this function is called before the loop is started, the loop count is 0. If this function is called while the loop activity is in process, the loop count is the number of times the loop ran. If this function is called after the loop is completed, the loop count is the total number of times the loop is defined to run.

Usage currentIteration = context.getLoopCount("1234567890");

Context.getProcessType()

The method returns the type of the request that triggers the custom workflow process.

Availability IBM Tivoli Identity Manager 4.x.

Synopsis context.getProcessType()

Returns String

Description

This member function returns the type of the request that triggers the custom workflow process. The function returns NEWACCOUNT or ACCOUNTDATACHANGE.

Usage if (context.getProcessType() == context.NEWACCOUNT) {...

Context.getRequestee()

The method returns the requestee associated with the request as a person object.

Availability IBM Tivoli Identity Manager 4.x.

Synopsis context.getRequestee();
Returns
A DirectoryObject that represents a Person.

Description
This member function returns the requestee associated with the request as a Person object. The requestee is the user who owns the associated, provisioned account.

Usage
requestee = context.getRequestee();

Context.getService()

The method returns the target service as a service entity object.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis
context.getService()

Returns
DirectoryObject

Description
This member function returns the target service as a Service entity object. The service entity is the service associated with the provisioned account.

Usage
service = context.getService();

Context.isAccountDataChanged()

The method identifies whether a specific account attribute was changed in the request that triggers the custom workflow process.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis
isAccountDataChanged(String attributeName)

Description
This member function identifies whether a specific account attribute was changed in the request that triggers the custom workflow process. If the request that triggers the custom workflow is NEWACCOUNT and the attribute is in the new account parameters, this function returns TRUE. Otherwise, this function returns FALSE. If the request that triggers the custom workflow is ACCOUNTDATACHANGE and the specified attribute is changed, this function returns TRUE. Otherwise, this function returns FALSE.

Usage
if (context.isAccountDataChanged("group")) {...

Credential

Credentials are associated with a shared access module operation, such as addCredentialToVault, checkin, or checkout.

Availability
IBM Security Identity Manager 6.0

Inherits from
DirectoryObject

Provided by
com.ibm.itim.script.extensions.model.CredentialModelExtension
Access mode

**EXCLUSIVE**
Indicates that an authorized user must access the credential through the checkout process.

**NON_EXCLUSIVE**
Indicates that an authorized user can access the credential without the checkout process.

**NON_SHARED**
Indicates that the credential is not intended for sharing.

Notification option

**NOTIFY_ONLY**
When the credential lease expires, a notification email is sent.

**NOTIFY_AND_CHECKIN**
When the credential lease expires, the credential is checked in automatically, and a notification email is sent.

Constructor

`new Credential(dn)`

Returns
The newly created Credential object that represents the credential with the specified DN, which is a String.

Methods

`getAccessMode()`
Returns an integer constant to represent the access mode, which can be EXCLUSIVE, NON_EXCLUSIVE, or NON_SHARED.

`getCheckoutDuration()`
Returns the maximum checkout time in hours.

`getNotificationRecipient()`
Returns the Participant object.

`getNotifyOption()`
Returns integer constant NOTIFY_ONLY, or NOTIFY_AND_CHECKIN.

`isCheckoutSearchEnable`
Returns true if the credential is enabled for search during checkout; returns false, otherwise.

`isNotifyOnly()`
Returns true if the system is configured to send only a notification when a lease is expired; returns false, otherwise.

`isPasswordViewable()`
Returns true if the credential password can be displayed to an authorized user; returns false, otherwise.

`isResetPasswordAtCheckin()`
Returns true if the credential password needs to be reset during the checkin process; returns false, otherwise.

**Credential.getAccessMode()**
The method returns the access mode of the credential.
Availability
IBM Security Identity Manager 6.0

Synopsis
Credential.getAccessMode()

Returns
Integer

Description
This function returns EXCLUSIVE, NON_EXCLUSIVE, or NON_SHARED.

Usage
var accessMode = credential.getAccessMode();
if (accessMode == Credential.EXCLUSIVE) {
  ...;
}

Credential.getCheckoutDuration()
The method returns the maximum checkout time for the credential in hours.

Availability
IBM Security Identity Manager 6.0

Synopsis
Credential.getCheckoutDuration()

Returns
Integer

Description
This function returns an integer value in hours.

Usage
var checkoutDuration = credential.getCheckoutDuration();

Credential.getNotifyOption()
The method returns the notification option when a credential lease is expired.

Availability
IBM Security Identity Manager 6.0

Synopsis
Credential.getNotifyOption()

Returns
Integer

Description
This function returns NOTIFY_ONLY or NOTIFY_AND_CHECKIN.

Usage
var notifyOption = credential.getNotifyOption();
if (notifyOption == Credential.NOTIFY_ONLY) {
  ...;
}

Credential.getNotificationRecipient()
The method returns the notification recipient when a credential lease is expired.

Availability
IBM Security Identity Manager 6.0

Synopsis
Credential.getNotificationRecipient()
Returns
Participant

Description
This function returns Participant object to whom the lease expiration email
is sent.

Note: The person who checked out the credential always gets a
notification when the lease is expired.

Usage
var participant = credential.getNotificationRecipient();

Credential.isCheckoutSearchEnable()
The method returns whether the credential is enabled for a checkout search.

Availability
IBM Security Identity Manager 6.0

Synopsis
Credential.isCheckoutSearchEnable()

Returns
Boolean

Description
This function returns true if the credential is enabled for a checkout search;
returns false otherwise.

Usage
var isSearchable = credential.isCheckoutSearchEnable();
if (isSearchable) {
    ...
}

Credential.isNotifyOnly()
The method returns whether the system must send only a notification email when
a credential lease is expired or not.

Availability
IBM Security Identity Manager 6.0

Synopsis
Credential.isNotifyOnly()

Returns
Boolean

Description
This function returns true if the notification option is NOTIFY_ONLY; returns
false if the notification option is NOTIFY_AND_CHECKIN.

Usage
var isNotifyOnly = credential.isNotifyOnly();
if (isNotifyOnly) {
    ...
}

Credential.isPasswordViewable()
The method returns whether the credential password can be displayed to an
authorized user or not.
Availability
IBM Security Identity Manager 6.0

Synopsis
Credential.isPasswordViewable()

Returns
Boolean

Description
This function returns true if the credential password can be displayed to an authorized user; returns false, otherwise.

Usage
var isDisplayPwd = credential.isPasswordViewable();
if (isDisplayPwd) {
    ...
}

Credential.isResetPasswordAtCheckin()
The method returns whether to reset the credential password during the checkin process or not.

Availability
IBM Security Identity Manager 6.0

Synopsis
Credential.isResetPasswordAtCheckin()

Returns
Boolean

Description
This function returns true if the credential password needs to be reset during the checkin process; returns false, otherwise.

Usage
var isResetPwd = credential.isResetPasswordAtCheckin();
if (isResetPwd) {
    ...
}

Delegate
The object provides the Delegate JavaScript object for use in the JavaScript environment of delegation notification. The Delegate JavaScript object and their use is described in this section.

Delegate
The Delegate object contains all the information associated with the current delegation operation.

Availability
IBM Tivoli Identity Manager 5.1.0.11
Delegation Notification context

Provided by
com.ibm.itim.script.extensions.DelegateExtension

Methods
Delegate.getDelegator()
    Returns the DirectoryObject that represents a system user such as
    the IBM Security Identity Manager account, whose activities are
    delegated.

Delegate.getDelegatee()
    Returns the DirectoryObject that represents a system user such as
    the IBM Security Identity Manager account, who is selected to be
    the delegate for the activities of the delegator.

Delegate.getStartDate()
    Returns a Date that contains the date and time when the delegation
    starts.

Delegate.getEndDate()
    Returns a Date that contains the date and time when the delegation
    ends.

Delegate.getRequester()
    Returns the DirectoryObject that represents a system user such as
    the IBM Security Identity Manager account, who initiated the
    delegation.

Description
    The Delegate object is available in the context of a delegation notification.
    The object retrieves the delegation information in the delegation
    notification template. The model script extensions are also available in the
    delegation notification context.

DirectoryObject

The object represents any IBM Security Identity Manager directory object or entity.

Availability
    IBM Tivoli Identity Manager 4.x

Constructor
    There is no specific constructor for this object. Specific constructors for
    Account, Person, Role, and Service return DirectoryObject.

    For example, new Service() returns a DirectoryObject.

Properties
    dn      String representing the distinguished name of the entity.
    name    String representing the logical name of the entity.
    profileName
        String representing the profile name of the entity.

Methods
    addProperty()
        Changes the value of the specified property, or adds the specified
        property if it does not exist. For multivalued objects,
        addProperty() adds the values to the specified property in the
        directory object and does not replace them.

    getChanges()
        Returns the changes made to the entity.

    getProperty()
        Returns the values of the property specified by the given name.
getPropertyNames()
   Returns a list of properties (attributes and relationships).

removeProperty()
   Removes the specified property.

setProperty()
   Changes the value of the specified property, or adds the specified
   property if it does not exist.

getPropertyAsDate()
   Returns the value of the specified property as a Date.

getPropertyAsString()
   Returns the value of the specified property as a String.

Description
This Object represents a IBM Security Identity Manager entity in the
JavaScript environment. Each IBM Security Identity Manager entity is
wrapped in one of these object classes.

**DirectoryObject.addProperty()**
The method adds or updates the value for the specified property.

**Availability**
IBM Tivoli Identity Manager 5.x

**Synopsis**
`directoryObject.addProperty(name, value)`

**Arguments**
- **name**  String representing the name of the property to be created
  or modified.
- **value**  The value to add to the property.

**Description**
This method changes the value of the specified property or adds the
specified property if it does not exist. This change is made locally to the
script environment, not to the data store. The value can be a single value
object or an array of objects. For multivalued objects, addProperty() adds
the values to the specified property in the directory object and does not
replace them. The value type (syntax) of object must be compatible with
the syntax of the specified property. This method is available for the
following data types:
- void addProperty(String name, Collection value);
- void addProperty(String name, Date value);
- void addProperty(String name, Map value);
- void addProperty(String name, boolean value);
- void addProperty(String name, byte value);
- void addProperty(String name, String value);
- void addProperty(String name, number value);
- void addProperty(String name, char value);

**Usage**
`directoryObject.addProperty("eruid", "jdoe");`

The getProperty method returns a Java array of objects that is stored in a
JavaScript JavaArray object. Unlike a standard JavaScript array, JavaArray
objects are used to access members of a Java array. Because Java arrays cannot be resized, the size of a JavaArray object cannot be changed. Also, JavaArray objects are typed. Setting a JavaArray element to the wrong type throws a JavaScript error.

In IBM Security Identity Manager, a JavaArray object cannot be passed directly back into a addProperty method. The JavaArray array might be converted into a standard JavaScript array as follows:

```javascript
jsAliases = new Array();
myPerson = person.get();
aliases = myPerson.getProperty("eraliases");
for (i=0; i < aliases.length; i++) {
  jsAliases[i] = aliases[i];
}
jsAliases[aliases.length] = "myNewAlias";
myPerson.addProperty("eraliases", jsAliases);
person.set(myPerson);
```

### DirectoryObject.dn

The field represents the distinguished name of the object.

**Availability**
IBM Tivoli Identity Manager 4.x.

**Synopsis**
`directoryObject.dn`

**Description**
This read-only field is a string that provides the distinguished name of the object. If the object holds information that was not created, there is no value.

**Usage**
`x = directoryObject.dn;`

### DirectoryObject.getChanges()

The method returns the changes made to the entity.

**Availability**
IBM Tivoli Identity Manager 4.x.

**Synopsis**
`directoryObject.getChanges()`

**Returns**
An array of change objects. If there are no changes, an empty array is returned. Each element in the array is an AttributeChangeOperation.

**Description**
This method returns the changes made to the entity. These changes are represented by change objects with the following members:

- **attr** String name of the attribute that is being changed.
- **op** An integer that identifies the type of change that is being made. The enumerated values are 1 for add, 2 for replace, and 3 for remove.
- **values** An array of objects that must be either added, removed, or replaced.

The changes are returned as an array of these change objects. If there are no changes, an empty array is returned.
Usage

```javascript
changes = directoryObject.getChanges();
for (i = 0; i < changes.length; i++) {
  name = changes[i].attr;
  if (changes[i].op == 1) {
  ...
  } else if (changes[i].op == 2) {
  ...
  } else {
  ...
  }
};
```

**DirectoryObject.getProperty()**

The method returns the values of the property specified by the given name.

**Availability**

IBM Tivoli Identity Manager 4.x

**Synopsis**

`directoryObject.getProperty(name)`

**Arguments**

- **name**  
  String representing the name of the property to return.

**Returns**

Either a String or a DirectoryObject. The type of object returned depends on the property obtained. If the specified property does not exist, an empty array is returned.

**Description**

This method returns the values of the property specified by the given name. The type of object returned depends on the property obtained. If the specified property does not exist, an empty array is returned.

The property name can be either an attribute name or a relationship name. For an attribute name, the return is a String[]. For a relationship name, an array of DirectoryObjects is returned. If an attribute and a relationship have the same name, then the attribute is returned. For example, an Account entity has both an owner attribute and an owner relationship.

**Usage**

When operating on an account, for example, the user ID property can return a String, where the owner property can return another entity (DirectoryObject). The owner entity can then be operated on with the `getProperty()` member to obtain information about it.

```javascript
userids = directoryObject.getProperty("eruid");
if (userids.length > 0)
  userid = userids[0];
owner = directoryObject.getProperty("owner");
if (owner.length > 0)
  ownerName = owner.getProperty("name")[0];
```

**Note:** These statements assume there is at least one value returned. If no values are returned, an array indexing violation occurs.

The `getProperty()` method returns a Java array of objects that is stored in a JavaScript `JavaArray` object. Unlike a standard JavaScript array, `JavaArray` objects are used to access members of a Java array. Since Java arrays cannot be resized, the size of a `JavaArray` object cannot be changed. Also, `JavaArray` objects are typed. Setting a `JavaArray` element to the wrong type throws a JavaScript error.
DirectoryObject.getPropertyAsDate()

The method returns the value of the property specified by the given name as a date object.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis
directoryObject.getPropertyAsDate(name)

Arguments
name  String representing the name of the property to return.

Returns
A Date object. If the specified property does not exist, current date is returned.

Description
This method returns the value of the property specified by the given name as a date object. If the specified property does not exist, current date is returned.

Usage
var createDate = directoryObject.getPropertyAsDate("ercreatedate");

DirectoryObject.getPropertyAsString()

The method returns the value of the property specified by the given name as a string.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis
directoryObject.getPropertyAsString(name)

Arguments
name  String representing the name of the property to return.

Returns
A String object. If the specified property does not exist, empty is returned. If the specified property has multiple values, only the first value is returned.

Description
This method returns the value of the property specified by the given name as a String object. If the specified property does not exist, empty string is returned. If the specified property has multiple values, only the first value is returned.

Usage
var name = directoryObject.getPropertyAsString("erservicename");

DirectoryObject.getPropertyNames()

The method returns a list of properties, such as attributes and relationships.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis
directoryObject.getPropertyNames()
Returns
An array of Strings.

Description
This method returns a list of properties as an array of Strings. A property can be either an attribute or a relationship.

Usage
properties = directoryObject.getPropertyNames();

**DirectoryObject.name**
The field represents the logical name of the object.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis
directoryObject.name

Description
This read-only field is a string that provides the logical name of the object, represented as a String. The physical attribute used as the name can be different for each type of object.

Usage
x = directoryObject.name;

**DirectoryObject.profileName**
The field returns the object profile name.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis
directoryObject.profileName()

Description
This read-only field is a string that provides the profile name of the object, represented as a String.

Usage
x = directoryObject.profileName;

**DirectoryObject.removeProperty()**
The method removes the property specified by the given name.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis
directoryObject.removeProperty(name)

Arguments

- name String representing the name of the property to remove.

Description
This method removes the specified property. This change is made locally to the script environment, not to the data store. The property name can be either an attribute name or a relationship name.

Usage
directoryObject.removeProperty("eruid");
**DirectoryObject.setProperty()**

The method sets the value of the specified property.

**Availability**
IBM Tivoli Identity Manager 4.x.

**Synopsis**

```javascript
directoryObject.setProperty(name, value)
```

**Arguments**

- `name`  String representing the name of the property to be created or modified.
- `value`  The value to set the property to.

**Description**

This method changes the value of the specified property, or adds the specified property if it does not exist. This change is made locally to the script environment, not to the data store. The value can be a single value object or an array of objects. The value type (syntax) of object must be compatible with the syntax of the specified property. This method is available for the following data types:

- void setProperty(String name, Collection value);
- void setProperty(String name, Date value);
- void setProperty(String name, Map value);
- void setProperty(String name, boolean value);
- void setProperty(String name, byte value);
- void setProperty(String name, String value);
- void setProperty(String name, number value);
- void setProperty(String name, char value);

**Usage**

```javascript
directoryObject.setProperty("eruid", "jdoe");
```

The `getProperty` method returns a Java array of objects that is stored in a JavaScript `JavaArray` object. Unlike a standard JavaScript array, `JavaArray` objects are used to access members of a Java array. Since Java arrays cannot be resized, the size of a `JavaArray` object cannot be changed. Also, `JavaArray` objects are typed. Setting a `JavaArray` element to the wrong type throws a JavaScript error.

In IBM Security Identity Manager, a `JavaArray` object cannot be passed directly back into a `setProperty` method. The `JavaArray` array into a standard JavaScript array as follows:

```javascript
jsAliases = new Array();
myPerson = person.get();
aliases = myPerson.getProperty("eraliases");
for (i=0; i < aliases.length; i++) {
    jsAliases[i] = aliases[i];
}
jsAliases[aliases.length] = "myNewAlias";

myPerson.setProperty("eraliases", jsAliases);
person.set(myPerson);
```

---

**EmailContext**

The object provides access to contextual information specific to a type of notification that is sent.
Some methods for accessing information change are based upon the listed notification types. (The Reminder/Approval/RFI/WorkOrder/ComplianceAlert Notification does not support this.)

- Activity Timeout Template
- Change Account Template
- Compliance Template
- New Account Template
- New Password Template
- Process Completion Template
- Process Timeout Template
- Restore Account Template
- Suspend Account Template

**Availability**

IBM Tivoli Identity Manager 4.6

**Provided by**

com.ibm.itim.workflow.script.EmailContextExtension

**Synopsis**

Call methods documented in this section as an `EmailContext` object. For example:

```java
notificationActivity=EmailContext.getActivity();
owner=EmailContext.getAccountOwnerName();
```

**Common methods**

These methods are available for all types of notifications:

**getActivity()**

Returns information about the most recent running activity. (Returns the `ActivityInfoOC` Java Object. To get the activity information in JavaScript object, use the object, `activity`).

**getActivity(java.lang.String actDefID)**

Returns information about the activity with the specified definition ID. (Returns the `ActivityInfoOC` Java Object.) This obtains information by using the Process.$dataName.get() workflow process. To get the activity information in JavaScript object, use `process.getActivity(java.lang.String actDefID)`.

**getParentProcess()**

Returns information about the parent process of the currently running process. (Returns the `ProcessInfoOC` Java object.) To get the process information of the parent process in JavaScript object, use `process.getParent()`.

**getProcess()**

Returns the information about the currently running process. (Returns the `ProcessInfoOC` Java object.) To get the process information of the parent process in JavaScript object, use `process.getProcess()`.

**getRootProcess()**

Returns information about the root process of the current running process. (Returns the `ProcessInfoOC` Java object.) To get the process information of the parent process in JavaScript object, use `process.getRootProcess()`. 
Account notification methods
These methods are available for all types of account notifications:

getAccountOwnerName()
    Returns the account owner name for the account.

getAccountServiceName()
    Returns the account service name for the account.

getAccountServiceProfileName()
    Returns the account service profile name for the account.

getAccountUserId()
    Returns the account user ID for the account.

hasNewAccess()
    Returns true if the account has new access and false otherwise.

hasRemovedAccess()
    Returns true if the account removed access and false otherwise.

getAccountNewAccessAsString()
    Returns String that contains list of new access separated by commas.

getAccountNewAccessList()
    Returns Array of String that contains the new access.

getAccountRemovedAccessAsString()
    Returns a string that contains the list of removed access separated by commas.

getAccountRemovedAccessList()
    Returns Array of String that contains the list of removed access.

Account Suspend/Deprovisioning Notification Methods:
These methods are only available for all types of account suspend/deprovision notifications:

getAction()
    Returns the action taken against the service (resource) itself.

getReason()
    Returns a descriptive reason for the deprovision.

Account New/Modify/Restore Notification Methods:
These methods are only available for all types of notifications for new, modified, and restored accounts:

showPassword()
    Returns whether to display the password when the user is notified of their new account.

getAccountPassword()
    Returns the account password for the account.

getPasswordExpirePeriod()
    Returns the password delivery expiration period.

getPasswordRetrievalUrl()
    Returns the password delivery URL in order to retrieve the password with the accounts shared secret.
**getTransactionId()**

Returns the password delivery transaction ID for picking up the password created for this account.

**Account Password Change Notification Methods:**

These methods are available for all types of account password change notifications:

**getAccountPassword()**

Returns the account password for the account.

**getPasswordExpirePeriod()**

Returns the password delivery expiration period.

**getPasswordRetrievalUrl()**

Returns the password delivery URL in order to retrieve the password with the accounts shared secret.

**getTransactionId()**

Returns the password delivery transaction ID for picking up the password created for this account.

---

**Enrole**

The object contains the general methods.

**Availability**

- All JavaScript contexts
- IBM Security Identity Manager Version 6.0
- IBM Tivoli Identity Manager Version 4.x

**Provided by**

com.ibm.itim.script.extensions.EnroleExtension

**Methods**

**generatePassword()**

Generates a password for a specific service.

**getAttributeValue()**

Get a single value attribute value.

**getAttributeValues()**

Get a multi-valued attribute value.

**localized()**

Localized message specified in &lt;Message&gt; XML format.

**log()**

Logs a message to the IBM Security Identity Manager log at ERROR level.

**logError()**

Logs the specified text to the IBM Security Identity Manager message log (msg.log) at ERROR level.

**logInfo()**

Logs the specified text to the IBM Security Identity Manager message log (msg.log) at INFO level.

**logWarning()**

Logs the specified text to the IBM Security Identity Manager message log (msg.log) at WARN level.
toGeneralizedTime()
Converts a time or date to generalized time format.

toMilleseconds()
Converts a String in generalized time format to an integer value in milliseconds.

traceMax()
Logs the specified text to the IBM Security Identity Manager trace log (trace.log) at DEBUG_MAX level.

traceMid()
Logs the specified text to the IBM Security Identity Manager trace log (trace.log) at DEBUG_MID level.

traceMin()
Logs the specified text to the IBM Security Identity Manager trace log (trace.log) at DEBUG_MIN level.

Description
Provides some common utilities for use in many different scripting contexts.

Enrole.generatePassword()
The method generates a new valid password for an account.

Availability
generatePassword() requires a service to work, so generatePassword() is only available when the ServiceExtension is used.

Synopsis
Enrole.generatePassword()

Returns
A String that is a valid password for the Service DirectoryObject stored in the "service" variable.

Description
This method generates a new valid password for a service.

Enrole.getAttributeValue()
The method retrieves the attribute's value.

Availability
Deprecated as of IBM Tivoli Identity Manager 4.3. Replace with DirectoryObject.getProperty()

Synopsis
Enrole.getAttributeValue(name, defaultValue)

Arguments

  name  String representing the name of the property to return.

  defaultValue  Default value to return if there is no value to return.

Returns
An Object. The type of object returned depends on the property obtained. If the specified property does not exist, the default value is returned.
Description
This method retrieves the value of the specified property.

**Enrole.getAttributeValues()**
The method retrieves a multi-valued attribute value.

**Availability**
Deprecated as of IBM Tivoli Identity Manager 4.3. Replace with DirectoryObject.getProperty().

**Synopsis**
Enrole.getAttributeValues(name)

**Arguments**
- name: String representing the name of the property to return.

**Returns**
An array of objects. The type of object returned depends on the property obtained. If the specified property does not exist, an empty array is returned.

Description
This method retrieves the value of the specified property.

**Enrole.localize()**
The method localizes a message specified in <Message> XML format.

**Availability**
IBM Tivoli Identity Manager Version 5.0

**Synopsis**
Enrole.localize(String xmlMsg, String localStr)

**Arguments**
- xmlMsg: A message specified in XML.
- localStr: A String that represents the locale to be used for globalization.

**Returns**
A localized message.

Description
This method localizes an XML message to the specified locale.

**Enrole.log()**
The method logs messages to the IBM Security Identity Manager message log (msg.log).

**Availability**
IBM Tivoli Identity Manager 4.6

**Synopsis**
Enrole.log(category, message);

**Arguments**
- category: The category of the log entry, entered as a String. The
category argument can be used or it can be left empty, but the argument must not be null.

**message**
The message to be logged, entered as a String.

**Description**
Logs a message to the IBM Security Identity Manager log at error level.

**Usage**
```
var roleDN = ..;(DN of role)
var role = new Role(roleDN);

// Put next statement on one line
Enrole.log("script", "The role name is " + role.getProperty("errolename")[0]);
```

Use the following new methods in IBM Security Identity Manager Version 6.0 to provide greater adaptability, control, or flexibility over the `Enrole.log()` method:
- `logError()`
- `logInfo()`
- `logWarning()`
- `traceMax()`
- `traceMid()`
- `traceMin()`

**Enrole.logError()**
The method logs text messages to the IBM Security Identity Manager message log (`msg.log`) with a message severity level of ERROR.

**Availability**
IBM Security Identity Manager Version 6.0

**Synopsis**
```
Enrole.logError(component, method, message);
```

**Arguments**

- **component**
The component of the log entry, entered as a String. The component can be any string. Logging can be controlled for components by setting specific log levels in the `enRoleLogging.properties` file.

- **method**
The string to display in the “Method” record of the message log. Useful to point where in the script the message originated.

- **message**
The string to represent the contents of the message log to be written to the log file.

**Description**
Writes an error message to the IBM Security Identity Manager message log (`msg.log`).
Usage
An example to write a msg.log message at ERROR level with the component name `com.ibm.myExtension` and the method name `postScriptOfAccountCreate`:

```javascript
var userName = "Joe"; // below is a single line
Enrole.logError("com.ibm.myExtension", "postScriptOfAccountCreate", "Recording error message after unsuccessful account creation for user " + userName + ");
```

**Enrole.logInfo()**

The method logs text messages to the IBM Security Identity Manager message log (msg.log) with a message severity level of INFO.

**Availability**
IBM Security Identity Manager Version 6.0

**Synopsis**

```javascript
Enrole.logInfo(component, method, message);
```

**Arguments**

- **component**
  The component of the log entry, entered as a String. The component can be any string. Logging can be controlled for components by setting specific log levels in the `enRoleLogging.properties` file.

- **method**
  The string to display in the “Method” record of the message log. Useful to point where in the script the message originated.

- **message**
  The string to represent the contents of the message log to be written to the log file.

**Description**

Writes an error message to the IBM Security Identity Manager message log (msg.log).

**Usage**

An example to write a msg.log message at INFO level with the component name `com.ibm.myExtension` and the method name `postScriptOfAccountCreate`:

```javascript
var userName = "Joe"; // below is a single line
Enrole.logInfo("com.ibm.myExtension", "postScriptOfAccountCreate", "Recording information message after account creation for user " + userName + ");
```

**Enrole.logWarning()**

The method logs text messages to the IBM Security Identity Manager message log (msg.log) with a message severity level of WARN.

**Availability**
IBM Security Identity Manager Version 6.0

**Synopsis**

```javascript
Enrole.logWarning(component, method, message);
```

**Arguments**

- **component**
  The component of the log entry, entered as a String. The component can be any string. Logging can be controlled...
for components by setting specific log levels in the
enRoleLogging.properties file.

**method**

The string to display in the “Method” record of the
message log. Useful to point where in the script the
message originated.

**message**

The string to represent the contents of the message log to
be written to the log file.

**Description**

Writes a warning message to the IBM Security Identity Manager message
log (msg.log).

**Usage**

An example to write a msg.log message at WARN level with the component
name com.ibm.myExtension and the method name
postScriptOfAccountCreate:

```javascript
var userName = "Joe";
// below is a single line
Enrole.logWarning("com.ibm.myExtension","postScriptOfAccountCreate",
"Recording warning message after account creation for user " + userName + ".");
```

**Enrole.toGeneralizedTime()**

The method converts a time or date to generalized time format.

**Availability**

IBM Tivoli Identity Manager 4.x.

**Synopsis**

`Enrole.toGeneralizedTime(time)`

**Arguments**

- **time**
  Integer time in milliseconds or a Date object.

**Description**

This method converts a time or date to generalized time format. Can be
used in either Identity Policies or in default entitlements.

**Usage**

```
seconds = Enrole.toGeneralizedTime(genTime);
```

**Enrole.toMilliseconds()**

The method converts a string in generalized time format to an integer value in
milliseconds.

**Availability**

IBM Tivoli Identity Manager 4.x.

**Synopsis**

`Enrole.toMilliseconds(genTime)`

**Arguments**

- **genTime**
  String in generalized time format.

**Description**

This method converts a String in generalized time format to an integer
value in milliseconds.

**Usage**

```
seconds = Enrole.toMilliseconds(genTime);
```
**Enrole.traceMax()**

The method logs text messages to the IBM Security Identity Manager trace log (trace.log) with a message severity level of DEBUG_MAX.

**Availability**
IBM Security Identity Manager Version 6.0

**Synopsis**
Enrole.traceMax((component, method, message);

**Arguments**
- **component**
  The component of the log entry, entered as a String. The component can be any string. Logging can be controlled for components by setting specific log levels in the enRoleLogging.properties file.

- **method**
  The string to display in the “Method” record of the message log. Useful to point where in the script the message originated.

- **message**
  The string to represent the contents of the trace message to be written to the log file.

**Description**
Wrote a DEBUG_MAX message to the IBM Security Identity Manager trace log (trace.log).

**Usage**
An example to write a trace.log message at DEBUG_MAX level with the component name com.ibm.myExtension and the method name postScriptOfAccountCreate:

```javascript
var userName = "Joe";
// below is a single line
Enrole.traceMax("com.ibm.myExtension","postScriptOfAccountCreate",
   "Recording DEBUG_MAX trace message after account creation for user " + userName + ".");
```

**Enrole.traceMid()**

Logs text messages to the IBM Security Identity Manager trace log (trace.log) with a message severity level of DEBUG_MID.

**Availability**
IBM Security Identity Manager Version 6.0

**Synopsis**
Enrole.traceMid((component, method, message);

**Arguments**
- **component**
  The component of the log entry, entered as a String. The component can be any string. Logging can be controlled for components by setting specific log levels in the enRoleLogging.properties file.

- **method**
  The string to display in the “Method” record of the message log. Useful to point where in the script the message originated.
message
The string to represent the contents of the trace message to be written to the log file.

Description
Writes a DEBUG_MID message to the IBM Security Identity Manager trace log (trace.log).

Usage
An example to write a trace.log message at DEBUG_MID level with the component name com.ibm.myExtension and the method name postScriptOfAccountCreate:

```javascript
var userName = "Joe";
// below is a single line
Enrole.traceMid("com.ibm.myExtension","postScriptOfAccountCreate",
"Recording DEBUG_MID trace message after account creation for user " + userName + ".");
```

Enrole.traceMin()

The method logs text messages to the IBM Security Identity Manager trace log (trace.log) with a message severity level of DEBUG_MIN.

Availability
IBM Security Identity Manager Version 6.0

Synopsis
Enrole.traceMin((component, method, message);

Arguments

component
The component of the log entry, entered as a String. The component can be any string. Logging can be controlled for components by setting specific log levels in the enRoleLogging.properties file.

method
The string to display in the “Method” record of the message log. Useful to point where in the script the message originated.

message
The string to represent the contents of the trace message to be written to the log file.

Description
Writes a DEBUG_MIN message to the IBM Security Identity Manager trace log (trace.log).

Usage
An example to write a trace.log message at DEBUG_MIN level with the component name com.ibm.myExtension and the method name postScriptOfAccountCreate:

```javascript
var userName = "Joe";
// below is a single line
Enrole.traceMid("com.ibm.myExtension","postScriptOfAccountCreate",
"Recording DEBUG_MIN trace message after account creation for user " + userName + ".");
```

Error

This object contains a script error description to notify the calling code of an exceptional runtime condition.

When an error is returned from a script evaluation, it is converted to a Java exception and thrown from the script evaluator class.
Availability
IBM Tivoli Identity Manager 4.6.x

Provided by
com.ibm.itim.script.extensions.EnroleExtension

Methods

setMessage()
Sets the message for the error.

getMessage()
Retrieves the error message for the error.

setErrorCode()
Sets the error code for the error.

getErrorCode()
Retrieves the error code for the error.

Usage
var sn = subject.getProperty("sn");
if(sn == null || sn.length == 0) {
    error.setMessage("sn was missing");
    return error;
} else {
    return sn[0];
}

Error.setMessage()
The method sets the message for the error.

Availability
IBM Tivoli Identity Manager 4.6.x

Synopsis
error.setMessage(String msg)

Arguments

msg  String representing the message to be set.

Description
This method sets the text for an error message. The function takes in one String parameter.

Usage  error.setMessage("sn was missing");

Error.getMessage()
The method retrieves the message set for an error.

Availability
IBM Tivoli Identity Manager 4.6.x.

Synopsis
error.getMessage()

Returns
String message for an error.

Description
This method retrieves the text of an error message.

Usage  messageValue = error.getMessage();
**Error.setErrorCode()**

The method sets the error code for the error.

**Availability**
IBM Tivoli Identity Manager 4.6.x.

**Synopsis**

```
error.setErrorCode(int code)
```

**Arguments**
- **code** Integer representing the error code.

**Description**
This method sets the error code for an error message. The function takes in one **Int** parameter.

**Usage**
```
error.setErrorCode(1);
```

**Error.getCodeError()**

The method retrieves the error code set for an error.

**Availability**
IBM Tivoli Identity Manager 4.6.x.

**Synopsis**

```
error.getCodeError()
```

**Returns**
- Integer value for an error code.

**Description**
This method retrieves the error code of an error message.

**Usage**
```
errorCodeValue = error.getCodeError();
```

**ExtendedPerson**

This object extends the Person object with the ownership type information for account adoption.

**Availability**
IBM Security Identity Manager Version 6.0.

**Inherited from**
- Person.

**Provided by**
- `com.ibm.tim.script.extensions.model.PersonModelExtension`

**Ownership type**
- **INDIVIDUAL**
  - String constant represents the default ownership type.

**Constructor**

```
new ExtendedPerson(dn)
```

**Arguments**
- **DN** DN string of a specific person entity.

**Returns**
- The new ExtendedPerson object that represents a person with the DN and INDIVIDUAL ownership type.
new ExtendedPerson(dn, ownershipType)

Arguments

DN
DN string of a specific person entity.

ownershipType
String representing one of the ownership types configured in IBM Security Identity Manager.

Returns
The new ExtendedPerson object that represents a person with the DN and ownership type. If the ownership type is invalid, it throws ScriptException.

new ExtendedPerson(person)

Arguments

person
Person object.

Returns
The new ExtendedPerson object that represents the person with the INDIVIDUAL ownership type.

new ExtendedPerson(person, ownershipType)

Arguments

person
Person object.

ownershipType
String representing one of the ownership types configured in IBM Security Identity Manager.

Returns
The new ExtendedPerson object that represents the person with the ownership type. If the ownership type is invalid, it throws ScriptException.

Methods

getOwnershipType()
Returns the ownership type.

setOwnershipType()
Sets the ownership type.

ExtendedPerson.getOwnershipType()
The method return the ownership type as a string.

Availability
IBM Security Identity Manager Version 6.0.

Synopsis
ExtendedPerson.getOwnershipType()

Returns
String.

Description
This method returns the ownership type.

Usage
var ownershipType = extendedPerson.getOwnershipType();
ExtendedPerson.setOwnershipType()
The method sets the value of the ownership type.

Availability
IBM Security Identity Manager Version 6.0.

Synopsis
ExtendedPerson.setOwnershipType(value)

Arguments
value  A string represents one of the ownership types configured in IBM Security Identity Manager.

Description
This method updates the ownership type. If the ownership type is invalid, it throws ScriptException.

Usage
var extendedPerson.setOwnershipType("System");

IdentityPolicy
The object represents the identity policy entity.

Availability
IBM Tivoli Identity Manager 4.x
Identity Policy context

Provided by
com.ibm.itim.policy.script.IdentityPolicyExtension

Methods
getNextCount()
Returns a number that can be appended to the end of a user name to make that user name unique.

getUserIDExists()
Checks if requested UID is already in use.

Description
This object represents a IBM Security Identity Manager Policy entity.

IdentityPolicy.getNextCount()
The method gets a number that can be appended to the end of a user name to make that user name unique. ServiceExtension must be loaded for getNextCount() to work.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis
IdentityPolicy.getNextCount(baseId)

Arguments
baseId  The base user name.
Returns
A number that can be appended to the end of a user name to make
the user name unique. Returns -1 if the user name is already
unique and -2 if an error occurs.

Description
This method checks whether requested UID is already in use.

Usage
num = IdentityPolicy.getNextCount(baseId);
return baseId + num;

IdentityPolicy.userIDExists()

The method checks if the requested UID is in use.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis
IdentityPolicy.userIDExists(uid, checkAllServices, checkRecycleBin)

Arguments
uid User identity.

checkAllServices
If set to true, all service instances are checked to see
whether the uid is used on an account of any service type.
If set to false, only the target service instance is checked.
This argument is optional. Default value is false.

checkRecycleBin
If set to true, the recycle bin is checked for any deleted
accounts. This parameter is intended to work in
conjunction with the checkAllServices parameter. Set this
parameter true only when the checkAllServices parameter
is also set to true. This argument is optional. Default value
is false.

Returns
True if the user ID exists, false otherwise.

Description
This method checks whether the requested UID is in use.

Usage
// To create a user ID without checking for it in the recycle bin but
// checking it against all services.
tf = IdentityPolicy.userIDExists("jason_jones", true, false);

PackagedApprovalDocument

A relevant data object used in multi-item approval, used exclusively in user
recertification workflows. This object is made up of multiple PackagedApprovalItem
objects from the user recertification approval and allows for searching and
retrieving recertification items.

Availability
IBM Tivoli Identity Manager 5.1.

Constructor
new PackagedApprovalDocument()
Constructs an empty approval document object. Instances might also be obtained in user recertification workflow and notifications by accessing the relevant data item "ApprovalDocument." For example, ApprovalDocument.get() returns a PackagedApprovalDocument in a user recertification workflow.

Properties

**TYPE_ACCOUNT**
A constant for approval items that are accounts.

**TYPE_GROUP**
A constant for approval items that are groups on other services but are not defined as an access.

**TYPE_GROUP_ACCESS**
A constant for approval items that are groups and also defined as accesses.

**TYPE_ITIM_GROUP**
A constant for approval items that are groups on services of type ITIM Service.

**TYPE_ROLE**
A constant for approval items that are roles.

Methods

**addItem(PackagedApprovalItem item)**
Returns a Boolean flag that indicates that a PackagedApprovalItem item is added in this approval document.

**containsDecisionCode(decisionCode)**
Returns a Boolean flag that indicates whether any of the items in this document that allow for decisions contain the specified decision code string. Valid decision codes are activity.APPROVED and activity.REJECTED.

**getDecisionItemCountByType(type)**
Returns the number of items in this document that support decisions and have the specified type. The types are defined as constants on this object, such as TYPE_ROLE or TYPE_ACCOUNT. This method considers all approval items in the document that supports decisions, including children of top-level items.

**getDecisionItemCountByType(type, includeChildren)**
Returns the number of items in this document that support decisions and have the specified type. The types are defined as constants on this object, such as TYPE_ROLE or TYPE_ACCOUNT. Depending on the value the includeChildren flag, this method might also count all items in this document, including any items that are children of the top-level items.

**getItemCountByType(type)**
Returns the number of items in this document that are of the specified type. The types are defined as constants on this object, such as TYPE_ROLE or TYPE_ACCOUNT. This method considers all approval items in the document, including children of top-level items.

**getItemCountByType(type, includeChildren)**
Returns the number of items in this document that are of the
specified type. The types are defined as constants on this object, such as TYPE_ROLE or TYPE_ACCOUNT. Depending on the value of the includeChildren flag, this method might also count all items in this document, including any items that are children of the top-level items.

getItemCountByTypeAndDecision(type, decisionCode)
Returns the number of items in this document that are of the specified type and that allow for decisions and contain the specified decision code string. The types are defined as constants on this object, such as TYPE_ROLE or TYPE_ACCOUNT. Valid decision codes are activity.APPROVED and activity.REJECTED. This method considers only top-level approval items and does not count the children of those items.

getItemsByType(type)
Returns the top-level items in this approval document that have the specified type as an array of PackagedApprovalItem objects. The types are defined as constants on this object, such as TYPE_ROLE or TYPE_ACCOUNT.

getItemsByTypeAndDecision(type, decisionCode)
Returns the top-level items in this approval document that have the specified type. If decisions are allowed, it contains the specified decision code string as an array of PackagedApprovalItem objects. The types are defined as constants on this object, such as TYPE_ROLE or TYPE_ACCOUNT. Valid decision codes are activity.APPROVED and activity.REJECTED.

removeItem(String identifier)
Returns a Boolean flag that indicates that a PackagedApprovalItem that corresponds to the identifier is removed from this approval document.

setDecisionCodeForAllItems(decisionCode)
Sets the specified decisionCode on all items in this document, including any children of top-level items. Any items that do not support decisions are skipped. Valid decision codes are activity.APPROVED and activity.REJECTED.

Description
The object represents the multi-item approval document in the JavaScript environment.

PackagedApprovalItem
A relevant data object used in IBM Security Identity Manager multi-item approval, used exclusively in user recertification workflows. This object represents the individual roles, accounts, and groups that are presented to the user during the recertification process. Some items might contain a decision code that indicates the choice of the approvers for that item. Each item also contains a list of children that is used to represent relationships between accounts and groups.

Availability
IBM Tivoli Identity Manager 5.1.

Constructor
new PackagedApprovalItem(itemType, value)
Constructs a PackagedApprovalItem object that does not support decisions and is read-only during the recertification approval activity. The parameters are an item type constant and value, where the value is a DirectoryObject that matches the type, such as Role or Account.

```javascript
new PackagedApprovalItem(itemType, value, decisionCode)
```

Constructs a PackagedApprovalItem object that supports decisions. The decisionCode parameter is either activity.APPROVED, activity.REJECTED, or null, where null indicates that a decision is required but not yet specified.

For example:

```javascript
new PackagedApprovalItem(PackagedApprovalDocument.TYPE_ACCOUNT, acctObj)
new PackagedApprovalItem(PackagedApprovalDocument.TYPE_ROLE, roleObj, activity.APPROVED)
```

**Properties**

**DECISION_NOT_APPLICABLE**
A constant for approval items that do not support decisions and are read-only during the recertification.

**Methods**

**getItemTypeString()**
Returns the type of the item, where the constant values are defined on the PackagedApprovalDocument object (TYPE_ROLE, TYPE_ACCOUNT, TYPE_GROUP, TYPE_GROUP_ACCESS).

**getDecisionCode()**
Returns the decision code for this item, where the possible values are activity.APPROVED and activity.REJECTED. This method might also return PackagedApprovalItem.DECISION_NOT_APPLICABLE if this item is for informational purposes only, or null if the decision is not yet specified.

**getValue()**
Returns a DirectoryObject for the role, account, or group of this item.

**getChildItems()**
Returns an array of PackagedApprovalItem objects that are the children of this item. For example, account items can have groups as their children.

**getChildItemsByDecision(decisionCode)**
Returns an array of PackagedApprovalItem objects that are the children of this item and have the specified decision code, such as activity.APPROVED or activity.REJECTED.

**Description**
The Object represents the IBM Security Identity Manager multi-item approval element in the JavaScript environment.

---

**Participant**

Workflow participant entity, which specifies an activity participant. In a mail node, this entity specifies the mail recipient.

Participant applies only to manual activity types, including Approval, RFI, WorkOrder, and Mail.
The participant of an activity can be specified during workflow design as Custom Defined Participant. In this case, the Participant JavaScript object can be used to construct the appropriate participant based on the process context.

**Availability**
IBM Tivoli Identity Manager 4.x

**Provided by**
com.ibm.itim.workflow.script.WorkflowExtension

**Constructor**
new Participant(type, dn)

**Arguments**
- **type** Code that categorizes the participant type.
- **dn** Optional DN of a specific entity.

**Returns**
The newly created and initialized participant object.

**Properties**
- **implementation**
  This property contains JavaScript that returns participant when the participant type is Custom.
- **name** Identifies the participant.
- **type** Code that categorizes the participant type.

**Description**
The participant specifies an activity participant. Participant applies only to manual activity types, including Approval, RFI, Work Order and Mail activities. The participant of an activity or recipient of a mail activity can be specified during workflow design as Custom Defined Participant. In this case, the Participant JavaScript object can be used to construct the appropriate participant based on the process context.

**Usage**
```javascript
//assume person is one of the relevant data in the workflow
//process for the target user involved
if( person.get().getProperty("title")[0]=="Manager" )
    return new Participant(ParticipantType.SYSTEM_ADMIN);
else
    return new Participant(ParticipantType.SUPERVISOR);

//assume person is one of the relevant data in the workflow
//process for the target user involved
if( person.get().getProperty("title")[0]=="Manager" )
    return new Participant(ParticipantType.USER, person.get().dn);
else
    ...
```

**Participant.implementation**
The field represents the custom defined participant.

**Availability**
IBM Tivoli Identity Manager 4.x.

**Synopsis**
`participant.implementation`
Description
This read-only field is a string that provides the custom-defined participant, which contains the JavaScript code to return the participant.

Usage  x = participant.implementation;

Participant.name
The field represents the DN of the participant.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis
participant.name

Description
This read-only field is a Distinguished Name that identifies the participant. It is only applicable to participant types of ROLE and USER.

Usage  x = participant.name;

Participant.type
the field represents the code that categorizes the participant type.

Availability
IBM Tivoli Identity Manager 4.x.

Synopsis
participant.type

Description
This read-only field is a string that represents a code that categorizes the participant type.

Usage  x = participant.type;

ParticipantType
An entity that represents the workflow participant type constants.

Availability
IBM Tivoli Identity Manager 4.x.

Provided by
com.ibm.itim.workflow.script.WorkflowExtension

Properties

DOMAIN_ADMIN
Participant type for the domain administrator of the organizational container. It is associated with the Subject account service (as specified by the Subject context in the workflow properties window).
participant = new Participant(ParticipantType.DOMAIN_ADMIN);

REQUESTOR
Participant type for the person that initiated the request. If a person initiates a change request for a person that triggers policy enforcement, the participant is the person that requests the change. For data loads, the participant is the system user. By setting the following property in $ISIM_HOME/data/enRole.properties to true,
an approval request that has the requester as the participant is automatically approved by the system:

```java
participant = new Participant(ParticipantType.REQUESTOR);
```

**REQUESTEE**
Participant type for the person designated as the requestee in the owner field of the relevant data.

```java
participant = new Participant(ParticipantType.REQUESTEE);
```

**ROLE**
Participant type for a specific organizational role. All user members of the role and its child roles are notified and are eligible to respond, the first response triggers the workflow to continue. In other words, specifying a role cannot be used to require multiple participants to approve the request.

```java
participant = new Participant(ParticipantType.ROLE, roleDN);
```

**ROLE_OWNER**
Participant type for the owner of the role (if specified). The Role is resolved based on the owners specified in the OrgRole listed as an input parameter for the operational workflow operation. If there is no OrgRole specified as an input parameter in the workflow, the participant is not resolved.

```java
participant = new Participant(ParticipantType.ROLE_OWNER);
```

**SERVICE_OWNER**
Participant type for the owner of the service (if specified). The Service is resolved based on the account object from the workflow relevant data that is marked as "Subject" in the properties window.

```java
participant = new Participant(ParticipantType.SERVICE_OWNER);
```

**SOD_POLICY_OWNER**
Participant type for the owners of the separation of duty policy (if specified). The owners are resolved based on the SeparationOfDutyRuleViolation object from the workflow relevant data that is marked as "Subject" in the properties window. If there is no SeparationOfDutyRuleViolation specified as the Subject of the workflow, the participant is not resolved.

The SOD_POLICY_OWNER participant type is used only in the approveSoDViolation global operation.

```java
participant = new Participant(ParticipantType.SOD_POLICY_OWNER);
```

**SPONSOR**
Participant type for the person designated as the sponsor with the sponsor relationship for the requestee (as marked in relevant data).

```java
participant = new Participant(ParticipantType.SPONSOR);
```

**SUPERVISOR**
Participant type for the supervisor or manager of the requestee. If none is specified for the requestee, then the supervisor designated on the organizational container of the requestee becomes the participant. If no supervisor is specified for the organizational container of the requestee, then the next level up is checked for a supervisor. The search continues up the tree until the top of the organization is reached. If no supervisor is found, the participant is unresolved.

```java
participant = new Participant(ParticipantType.SUPERVISOR);
```
SYSTEM_ADMIN
   Participant type for a member of the IBM Security Identity
   Manager System Administrator group.
   participant = new Participant(ParticipantType.SYSTEM_ADMIN);

USER     Participant type for a specific person to respond to the request. The
          person must have a IBM Security Identity Manager account.
          participant = new Participant(ParticipantType.USER, userDN);

ITIM GROUP Participant type for a specific ITIM group. Though all members of
                  the group are notified, and all are eligible to respond, the first
                  response triggers the workflow to continue. Specifying a group
                  cannot be used to require multiple participants to approve the
                  request.
                  participant = new Participant(ParticipantType.GROUP, groupDN);

Description
   This entity represents the workflow participant type constants.

Person

The object represents the person entity.

Availability
   IBM Tivoli Identity Manager 4.x.

Provided by
   com.ibm.itim.script.extensions.model.PersonModelExtension

Inherits From
   DirectoryObject

Constructors
   new Person(String dn)
      Arguments:
      dn      Optional DN of a specific entity.

   new Person(DirectoryObject directoryObject)
      Arguments:
      directoryObject
         DirectoryObject to be contained in the person

   new Person(DirectoryObjectEntity directoryObjectEntity)
      Arguments:
      directoryObjectEntity
         DirectoryObjectEntity to be contained in the person

Methods
   getAllAssignmentAttributes()
      Returns an array of the RoleAssignmentAttribute objects that are
      defined in all of authorized roles for this person. The authorized
      roles consist of both the direct roles for this person and also all of
      the parent roles of the direct roles.

   getRoleAssignmentData()
      Returns all role assignment data for the person.
getRoleAssignmentData(String roleAssignedDN)
   Returns all role assignment data for the person for the specified role.

getRoles()
   Returns an array of DirectoryObjects, each representing a role.

getNewRoles()
   Returns an array of newly added roles for the person.

getRemovedRoles()
   Returns an array of removed roles for the person.

isInRole(String roleName)
   Determines whether the person belongs to the role. Returns Boolean.

removeRole()
   Removes the person from the specified role.

removeRoleAssignmentData(String roleAssignedDN)
   Removes all role assignment data for the person from the specified role.

updateRoleAssignmentData(RoleAssignmentObject[] roleAssignmentObject)
   Updates a person with the role assignment attribute value changes that are defined in the set of RoleAssignmentObjects.

Person.getAllAssignmentAttributes()
The method returns an array of the RoleAssignmentAttribute objects that are defined for all of authorized roles for this person. The authorized roles consist of both the direct roles for this person and also all the parent roles of the direct roles.

Availability
   IBM Security Identity Manager 6.0

Synopsis
   person.getAllAssignmentAttributes()

Arguments
   None

Description
   This method is defined on the Person object. It returns an array of the RoleAssignmentAttribute objects that are defined in all of authorized roles for this person. The authorized roles consist of both the direct roles for this person and also all the parent roles of the direct roles. The method returns an empty array if no assignment attribute exists. RoleAssignmentAttribute objects contains role assignment attribute name, role name, and role DN.

Usage
   //The script is used in a workflow, in which Entity is a person object.
   var person = Entity.get();

   //get assignment attributes of the person
   var attributeList = person.getAllAssignmentAttributes();
   if (attributeList.length == 0) {
       Enrole.log("script", "No assignment attribute for this role: "+role.name);
       return;
   }
// print out the role assignment attribute name.
for (var i=0; i < attributeList.length; i++) {
    var roleAtr = attributeList[i];
    Enrole.log("script","attribute name-----: " + roleAtr.getName());
}

Person.getRoleAssignmentData()

The method returns all the role assignment data for the person, as an array of RoleAssignmentObject objects that contain the role assignment values, defined Role DN and assigned Role DN.

Availability
IBM Security Identity Manager 6.0

Synopsis
person.getRoleAssignmentData()

Arguments
none

Description
This method is defined on the Person object. It returns an array of RoleAssignmentObject objects, containing the role assignment values, defined Role DN, and assigned Role DN. The method returns an empty array if no assignment data exists.

Usage
//The script is used in a workflow, in which Entity is a person object.
var person = Entity.get();
var assignmentObjects = person.getRoleAssignmentData();
if (assignmentObjects.length == 0) {
    Enrole.log("script", "There is no assignment values for " + person.name);
    return;
}
var str = "The number of role assignment objects returned from
person.getRoleAssignmentData(): " +
    assignmentObjects.length + "\n";
for(var i=0; i<assignmentObjects.length; i++) {
    var obj = assignmentObjects[i];
    str += obj.toString() + "\n";
}
Enrole.log("script", "The assignment attribute data for person:"+
    person.name+" is:"+ str);

Person.getRoleAssignmentData(String roleAssignedDN)

The method returns all the role assignment data for the person. The data is an array of RoleAssignmentObject objects that contain the role assignment values, defined Role DN, and assigned Role DN for the specified assigned role.

Availability
IBM Security Identity Manager 6.0

Synopsis
person.getRoleAssignmentData(String roleAssignedDN)

Arguments
roleAssignedDN
The distinguished name of the assigned role

Description
This method is defined on the Person object. It returns an array of RoleAssignmentObject objects, containing the role assignment values,
defined Role DN, and assigned Role DN for a specified assigned role. The method returns an empty array if no assignment data exists.

Usage

// The script is used in a workflow, in which Entity is a person object.
var person = Entity.get();
var roleDNs = person.getProperty("erroles");
if(roleDNs.length == 0) {
    Enrole.log("script", person.name + " does not have any role");
    return;
}

// Get role assignment data for the first role.
var roleDN = roleDNs[0];
var role = new Role(roleDN);
var assignmentObjects = person.getRoleAssignmentData(roleDNs[0]);
if (assignmentObjects.length == 0) {
    Enrole.log("script", person.name + " does not have any assignment objects for role: " + role.name);
    return;
}

var str = "The number of role assignment objects returned from person.getRoleAssignmentData() for " + role.name + " : " + assignmentObjects.length + "\n";
for(var i=0; i<assignmentObjects.length; i++) {
    var obj = assignmentObjects[i];
    str += obj.toString() + "\n";
}
Enrole.log("script", str);

Person.getRoles()

The method returns roles assigned to a Person.

Availability
IBM Tivoli Identity Manager 4.6

Synopsis

person.getRoles()

Description
This method defined on the Person object returns an array of roles that the person belongs to. The return type is an array of entities, which are instances of role directory entity objects. The properties available on the Entity Objects are name and description.

Usage

// logs the names of all roles that a person belongs to
var per = person.get();
var rolesArray = per.getRoles();
if(rolesArray.length>0){
    Enrole.log("script", per.getProperty("cn")[0] + " belongs to following roles: ");
    for( var i=0; i<rolesArray.length; i++) {
        Enrole.log("script", rolesArray[i].getProperty("errolename")[0]);
    }
} else {  
    Enrole.log("script", per.getProperty("cn")[0] + " does not belong to any roles");
}

Person.getNewRoles()

The method returns an array of newly added static roles for a Person.
Availability
IBM Tivoli Identity Manager 5.0

Synopsis
`person.getNewRoles()`

Description
This method defined on the person object returns an array of new static roles associated with the person. The return type is an array of `DirectoryObjects`.

Note: The person object is often a runtime object in memory, and these new static roles were not added to the directory.

Usage
```
var newRoles = per.getNewRoles();
```

**Person.getRemovedRoles()**
The method returns an array of removed static roles for the Person.

Availability
IBM Tivoli Identity Manager 5.0

Synopsis
`person.getRemovedRoles()`

Description
This method defined on the person object returns an array of static roles from which the person was removed. The return type is an array of `DirectoryObjects`.

Note: The person object is often a runtime object in memory, and these static roles were not removed from the directory.

Usage
```
var removedRoles = per.getRemovedRoles();
```

**Person.isInRole()**
The method evaluates whether a Person belongs to a role.

Availability
IBM Tivoli Identity Manager 4.6

Synopsis
`person.isInRole(roleName)`

Arguments
- `roleName`
The name of the role to check.

Description
Given a person object and the name of the role, determine whether the person belongs to the role. If the role is not uniquely determined by the `roleName` parameter or if the person cannot be found, then return an error object.

Usage
```
// Check whether the person is in the role Manager and log a message
var per=person.get();
```
if(!per.isInRole("Manager")) {
    Enrole.log("script",per.getProperty("cn")[0] +
    "does not belong to role Manager");
} else {
    Enrole.log("script",per.getProperty("cn")[0] +
    "belong to role Manager");
}

**Person.removeRole()**

The method removes the person from the specified role.

**Availability**
IBM Tivoli Identity Manager 5.0

**Synopsis**

```
person.removeRole(role)
```

**Arguments**

- **role**  
  Role object that represents the role from which the person is removed.

**Description**

Removes the person from the role.

**Note:** This operation removes only the role from the Person object in run time, and it does not remove the role from the directory.

**Usage**

```
//Remove the first role in the Person object
var roles = person.getRoles();
if (roles.length > 0) {
    person.removeRole(roles[0]);
}
```

**Person.removeRoleAssignmentData()**

The method removes all role assignment data of the person for an array of assigned Roles. It does not directly change data in the data source, but removes from memory the data inside the person object.

**Availability**
IBM Security Identity Manager 6.0

**Synopsis**

```
person.removeRoleAssignmentData(String [] roleAssignedDNs)
```

**Arguments**

- **roleAssignedDNs**  
  An array of distinguished names of the assigned role.

**Description**

This method is defined on the Person object. It removes all role assignment data of the person for an array of assigned roles.

**Usage**

```
//The script is used in a workflow, in which Entity is a person object.
var person = Entity.get();
var roleDNs = person.getProperty("erroles");
if(roleDNs.length == 0) {
    Enrole.log("script", person.name + " does not have any roles");
    return;
}
//remove the role assignment attribute.
person.removeRoleAssignmentData(roleDNs);

**Person.updateRoleAssignmentData()**

The method updates a person with the role assignment attribute value changes that are defined in the set of RoleAssignmentObjects. It does not directly change data in the data source, but updates (in memory) the data inside the person object.

**Availability**
IBM Security Identity Manager 6.0

**Synopsis**

```javascript
person.updateRoleAssignmentData(RoleAssignmentObject [] roleAssignmentObject)
```

**Arguments**

`roleAssignmentObject`
A list of RoleAssignmentObjects that contains the role assignment attribute value change set to be applied.

**Description**

This method is defined on the Person object. It updates a person with the role assignment attribute value changes that are defined in the set of RoleAssignmentObjects.

**Usage**

```javascript
//The script is used in a workflow, in which Entity is a person object.
var person = Entity.get();
var roleDNs = person.getProperty("erroles");
if(roleDNs.length == 0) {
  Enrole.log("script", person.name + " does not have any role");
  return;
}

//construct a new RoleAssignmentObject
var assignmentObj = new RoleAssignmentObject(roleDNs[0], roleDNs[0]);
assignmentObj.addProperty("attr_3", ["newv1", "newv2"]);
person.updateRoleAssignmentData([assignmentObj]);
```

---

**PersonSearch**

The object searches for a person.

**Availability**

IBM Tivoli Identity Manager 4.x
Provisioning Policy context
Service Selection Policy context

**Provided by**

com.ibm.itim.script.extensions.model.PersonModelExtension

**Constructor**

```javascript
new PersonSearch()
```

**Returns**

The newly created and initialized person search object.

**Methods**

`searchByFilter()`

Search for a person by a filter.
searchByURI()
Search for a person by URI in an organizational container.

Description
The entity implements the IBM Security Identity Manager PersonSearch class. The API Javadoc for this class is in the following directory:
$ISIM_HOME/extensions/version_number/api/com/ibm/itim/dataservices/model/domain/

PersonSearch.searchByFilter()
The method searches for a person by a filter.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis
personSearch.searchByFilter(profileName, filter, scope)

Arguments

  profileName
  The name of the person profile to use.

  filter
  LDAP search filter that defines the criteria for returned containers to meet. The filter must be in the format defined by RFC2254.

  scope
  Optional search scope. Use 1 for One Level Scope and 2 for SubTree Scope. One Level Scope is the default scope.

Returns
An array of DirectoryObjects representing the results of the search.

Description
This method searches for a person by a filter.

Usage
var personSearch = new PersonSearch();
var searchResult1 = personSearch.searchByFilter("Person", "(sn=Smith)", 2);

// use default one level scope
var searchResult2 = personSearch.searchByFilter("Person", "(sn=Smith)");

PersonSearch.searchByURI()
The method finds a person by URI within an organizational container.

Availability
IBM Security Identity Manager 6.0

Synopsis
PersonSearch.searchByURI(containerDN, uri)

Arguments

  Container DN
  String representing the distinguished name of the parent organizational container.

  uri
  String representing the URI of the person.
Returns
A Person object.

Description
Given the distinguished name of the parent organizational container and the person URI, this method finds the person. If the person is not found, this function returns null. If more than one persons found, this function throws a scripting exception.

Usage
```javascript
var person= (new PersonSearch()).searchByURI(container.dn, uri);
if (person != null) {
    Enrole.log("script", "Found " + person.getProperty("cn") );
}
```

---

**PostOffice**

The object post office object that consolidates notifications.

Availability
IBM Tivoli Identity Manager 4.6.x

Provided by
com.ibm.itim.mail.postoffice.script.PostOfficeExtension

Methods

- `getAllEmailMessages()`
  Obtains the Subject, Text Body, and HTML Body of each individual message contained in an aggregate message.

- `getEmailAddress()`
  Contains the email address that is the destination of the aggregate email message.

- `getPersonByEmailAddress()`
  Returns the Person that corresponds to the email address specified.

- `getTopic()`
  Returns the topic of the aggregated email message.

The `getAllEmailMessages()` extension allows access to the NotificationMessage object. Do not call the `getHtmlMessage()` method from a template. This call returns an XHTML version of the notification text. It is not possible to embed XML documents, so a call to this method results in a template execution failure. Use the text body of the original notifications by calling `getMessage()` instead.

---

**PostOffice.getAllEmailMessages()**

The message returns an array of NotificationMessage objects.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis
PostOffice.getAllEmailMessages()

Description
This JavaScript extension returns an array of NotificationMessage objects for obtaining the Subject, Text Body, and HTML Body of each message in an aggregate message.

Usage
An example of how to iterate through the returned array in JavaScript is as follows:
Here are the email text bodies fetched using the JavaScript extension:

```javascript
var msgListIterator = PostOffice.getAllEmailMessages().iterator();
var returnString = "<br/>
while (msgListIterator.hasNext()) {
    returnString = returnString + msgListIterator.next().getMessage() + "<br/>";
}
return returnString;
```

**PostOffice.getEmailAddress()**

The method returns email address of aggregate email destination.

**Availability**
IBM Tivoli Identity Manager 4.x

**Synopsis**
PostOffice.getEmailAddress()

**Description**
This JavaScript extension returns a String containing the email address that is the destination of the aggregate email message.

**Usage**
destinationAddress = PostOffice.getEmailAddress();

**PostOffice.getPersonByEmailAddress()**

The method returns the Person object that corresponds to this email address.

**Availability**
IBM Tivoli Identity Manager 4.x

**Synopsis**
PostOffice.getPersonByEmailAddress(String email)

**Description**
This JavaScript extension returns the Person object that corresponds to the email address specified.

**Usage**
targetPerson = PostOffice.getPersonByEmailAddress();

Examples:
targetPerson = PostOffice.getPersonByEmailAddress("user@itim.com");
targetPerson = PostOffice.getPersonByEmailAddress(PostOffice.getEmailAddress());

**PostOffice.getTopic()**

The method returns the topic string of the aggregate email.

**Availability**
IBM Tivoli Identity Manager 4.x

**Synopsis**
PostOffice.getTopic()

**Description**
This JavaScript extension returns a string containing the topic of the aggregated email message.

**Usage**
topicString = PostOffice.getTopic();
Process

Represents the IBM Security Identity Manager workflow process.

Availability
IBM Tivoli Identity Manager 4.x

Provided by
The Process JavaScript Object in the WorkflowExtension returns a Process object. The object represents the current workflow process. The parent processes of the current workflow can be returned by calling Process.getParent() recursively, and the parent process is also a Process object.

Properties

Note: Custom result codes are supported in the workflow designer for approval activities.

APPROVED
Approved process summary code. Result code is AA.

ESCALATED
Escalated process summary code. Result code is ES.

FAILED
Failed process summary code. Result code is SF.

PARTICIPANT_RESOLVE_FAILED
Participant resolved failure process summary code. Result code is PF.

PENDING
Pending process summary code. Result code is PE.

REJECTED
Rejected process summary code. Result code is AR.

SUBMITTED
Submitted process summary code. Result code is RS.

SUCCESS
Success process summary code. Result code is SS.

TIMEOUT
Time out process summary code. Result code is ST.

WARNING
Warning process summary code. Result code is SW.

comment
Provides additional information about the process given when defined in the workflow designer.

description
Describes the purpose of the process given when defined in the workflow designer.

id
Assigned by the workflow designer to uniquely identify the workflow process within the workflow engine.

name
Label given this activity when defined in the workflow designer.
parentld
Uniquely identifies the parent process (if any) that started this process.

requesteeDN
Uniquely identifies the requestee if the requestee is a user in the IBM Security Identity Manager data store.

requesteeName
Name of the process requestee.

requestorName
The name of the process requestor if the requestor is a user.

requestorType
Categorize the requestor

resultDetail
An application-specific string that provides more detail about the result of the process.

resultSummary
An application-specific string that represents the summary result of the process.

started
Indicates when the process started.

state
Code that represents the current state of the process.

subject
Describes the object that is the focal point of the workflow process.

type
Code that categorizes the process given when defined in the workflow designer.

Methods

auditEvent()
Create an event in the audit trail specific to the activity.

getActivity()
Returns an activity with the ID and index.

getParent()
Get the parent process (if any) that started this process.

getRootProcess()
Returns the JavaScript Process object that contains information about the root process.

getRootRequesterName()
Returns String of requester name of the root process.

setRequesteeData()
Change the requestee data for the current process.

setResult()
Change the result member of the activity in the current activity.

setSubjectData()
Change the subject data for the current process.

Description
This entity represents the current workflow process is running.
**Process.auditEvent()**

The method creates an event in the audit trail.

**Availability**
IBM Tivoli Identity Manager 4.x

**Synopsis**
`process.auditEvent(event)`

**Arguments**
- `event` String representing the event to be audited.

**Description**
This method creates an event in the audit trail specific to the process. The function takes in one parameter that can be any JavaScript object that can be translated into a string for storage. In the audit trail, the event is automatically time stamped.

**Usage**
```
process.auditEvent("Task completed");
```

**Process.comment**

The field provides additional information about the process.

**Availability**
IBM Tivoli Identity Manager 4.x

**Synopsis**
`process.comment`

**Description**
This read-only field is a string that provides additional information about the process given when defined in the workflow designer.

**Usage**
```
x = process.comment;
```

**Process.description**

The field represents the purpose of the process.

**Availability**
IBM Tivoli Identity Manager 4.x

**Synopsis**
`process.description`

**Description**
This read-only field is a string that describes the purpose of the process when defined in the workflow designer.

**Usage**
```
x = process.description;
```

**Process.getActivity()**

The method returns an activity with the ID and index.

**Availability**
IBM Tivoli Identity Manager 4.x

**Synopsis**
`process.getActivity(id, index)`

**Arguments**
- `id` Activity ID assigned by the workflow designer.
index  Optionally identifies specific activity if there is more than one activity with the ID.

Returns
The associated Activity.

Description
This method returns an activity with the ID and index in the event that there is more than one activity with the ID. This might occur if the activity of the given ID is called multiple times in a loop in the workflow process. If there is no activity with the ID and index, this function returns null. If the optional index is not specified and if there is more than one activity with the ID, the first activity with the ID is returned.

Usage
```java
theFirstActivity = process.getActivity("id1", 3);
theActivityName = theFirstActivity.name;
theSecondActivity = process.getActivity("id2");
theActivityName = theSecondActivity.name;
```

Process.getParent()

The method returns the parent process (if any) that started this process.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis
`process.getParent()`

Returns
The parent Process. If there is no parent, a null is returned.

Description
This method returns the parent process (if any) that started this process.

Usage
```java
parent = process.getParent();
parentName = parent.name;
```

Process.getRootProcess()

The method returns the root process (if any) that started this process.

Availability
IBM Tivoli Identity Manager 5.0

Synopsis
`process.getRootProcess()`

Returns
The root process. If there is no root process, a null is returned.

Description
This method returns the root process (if any) of this process.
Usage
root = process.getRootProcess();
rootName = root.name;

Process.getRootRequesterName()
The method returns the root requester name.

Availability
IBM Tivoli Identity Manager 4.6

Synopsis
process.getRootRequesterName()

Description
This method returns the root requester name of the workflow process
initiator.

Usage
rootRequester = process.getRootRequesterName();

Process.guid
The generated unique identifier assigned to the process at runtime.

Availability
IBM Tivoli Identity Manager 5.x

Synopsis
process.guid

Description
This read-only field is a String of the generated unique identifier for the
workflow process in the workflow engine.

Usage
x = process.guid;

Process.id
The generated unique identifier assigned to the process at runtime.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis
process.id

Description
This read-only field is a string of the generated unique identifier for the
workflow process in the workflow engine.

Usage
x = process.id;

Process.name
The label assigned to the process.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis
process.name

Description
This read-only field is a string assigned by the workflow designer to label
this process.
**Usage**  
\[ x = \text{process.name}; \]

**Process.parentId**

The field uniquely identifies the parent process that started this process.

**Availability**

IBM Tivoli Identity Manager 4.x

**Synopsis**

\[ \text{process.parentId} \]

**Description**

This read-only field is a string representation of the long integer that uniquely identifies the parent process (if any) that started this process.

**Usage**  
\[ x = \text{process.parentId}; \]

**Process.requesteeDN**

The field uniquely identifies the requestee if the requestee is a user in the IBM Security Identity Manager data store.

**Availability**

IBM Tivoli Identity Manager 4.x

**Synopsis**

\[ \text{process.requesteeDN} \]

**Description**

This read-only field is a string that uniquely identifies the requestee if the requestee is a user in the IBM Security Identity Manager data store. Not all requestees are users (that is, the process can act on a policy, not a user directly), so this member can be empty.

**Usage**  
\[ x = \text{process.requesteeDN}; \]

**Process.requestorDN**

The field specifies the distinguished name of the process requester, if the requester is a user in the IBM Security Identity Manager data store.

**Availability**

IBM Tivoli Identity Manager 4.x

**Synopsis**

\[ \text{process.requestorDN} \]

**Description**

This read-only field is a string that represents the distinguished name of the process requester. This string is displayed only if the requester is a user in the IBM Security Identity Manager data store. Not all requesters are users (that is, the process can act on a policy, not a user directly), so this member can be empty.

**Usage**

\[
\begin{align*}
&\text{if (process.requestorType == "U")} \\
&x = \text{process.requestorDN};
\end{align*}
\]

**Process.requesteeName**

The field represents the name of the process requestee as a string.

**Availability**

IBM Tivoli Identity Manager 4.x
Synopsis

process.requesteeName

Description
This read-only field is a string that provides the name the requestee if the
requestee is a user in the IBM Security Identity Manager data store. Not all
requestees are users (that is, the process can act on a policy, not a user
directly), so this member can be empty.

Usage  x = process.requesteeName;

Process.requestorName

The field represents the name of the process requester if the requester is a user.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis
process.requestorName

Description
This read-only field is a string that represents the name of the process
requester if the requester is a user.

Usage
if (process.requestorType == "U")
   x = process.requestorName;

Process.requestorType

The field categorize the requestor.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis
process.requestorType

Description
This read-only field is a string that categorizes the requestor. The potential
categories, or types, are:

- U for user
- S for the workflow engine
- P for the system

Usage
x = process.requestorType;
if (x == "U")
    ...
else if (x == "S")
    ...
else if (x == "P")
    ...

Process.resultDetail

The field details about the result of the process.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis
process.resultDetail
**Description**

This read-only field is an application-specific string that provides more detail about the result of the process.

**Usage**

x = `process.resultDetail;`  

**Process.resultSummary**

The field represents the summary of the result of the process.

**Availability**

IBM Tivoli Identity Manager 4.x

**Description**

This read-only field is an application-specific string that provides a summary of the result of the process.

**Usage**

x = `process.resultSummary;`  

**Process.setRequesteeData()**

The method changes the requestee data for the current process.

**Availability**

IBM Tivoli Identity Manager 4.x

**Synopsis**

`process.setRequesteeData(person)`

**Arguments**

- `person`  
  DirectoryObject representing the new requestee.

**Description**

This method changes the requestee data for the current process. It is not supported for a process that is not the current process. It not only updates the current process in the script, but also in the workflow engine. The `requesteeData` argument contains a person distinguished name or a collection of strings from which the requestee data can be extracted.

**Usage**

`process.setRequesteeData(person);`  

**Process.setResult()**

The method changes the result member of the process.

**Availability**

IBM Tivoli Identity Manager 4.x

**Synopsis**

`process.setResult(summary, detail)`

**Arguments**

- `summary`  
  String code that represents the result summary.

- `detail`  
  String representing the result details.

**Description**

This method changes the result member of the process in the current process. It is supported for current activities in the current workflow process. The result is composed by an application-specific summary code, and optional more detailed application-specific description. The summary
code can indicate a success or failure. This summary code is stored as the 
resultSummary member locally and updated in the relevant data in the 
workflow engine. The detail is stored as the resultDetail member locally 
and updated in the relevant data in the workflow engine.

Usage

process.setResult(process.FAILED, "Unable to connect to resource");

Process.setSubjectData()
The method changes the subject data for the current process.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis

process.setSubjectData(person)

Arguments

person
DirectoryObject representing the new subject.

Description
This method changes the subject data for the current process. It is not 
supported for a process that is not the current process. It not only updates 
the current process in the script, but also in the workflow engine. The 
subjectData argument contains a person distinguished name or a 
collection of strings from which the subject data can be extracted.

Usage

process.setSubjectData(person);

Process.started
The field represents the JavaScript date that indicates when the process started.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis

process.started

Description
This read-only field is code string that represents the JavaScript Date that 
indicates when the process started.

Usage

x = process.started;

Process.state
The field represents the current state of the process.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis

process.state

Description
This read-only field is code string that represents the current state of the 
process. The state can have the following values:
• R for running
• I for not started
• T for terminated
• A for aborted
• S for suspended
• C for completed
• B for bypassed

Usage
```java
if (process.state == "S") {
...
}
```

**Process.subject**

The field represents the object that is the focal point of the workflow process.

**Availability**
IBM Tivoli Identity Manager 4.x

**Synopsis**
```
process.subject
```

**Description**
This read-only field is code string that describes the object that is the focal point of the workflow process. This string can be an identity in the system, an account, a policy, or another object.

Usage
```
x = process.subject;
```

**Process.type**

The field represents the type of process.

**Availability**
IBM Tivoli Identity Manager 4.x

**Synopsis**
```
process.type
```

**Description**
This read-only field is code string that categorizes the process when defined in the workflow designer.

Usage
```
x = process.type;
```

**ProcessData**

The object represents the workflow process data entity.

**Availability**
IBM Tivoli Identity Manager 4.x
Workflow context

**Provided by**
com.ibm.itim.workflow.script.WorkflowExtension

**Methods**
- get() Returns a JavaScript object that represents the value of the relevant data item.
- set() Changes the value of the relevant data item.
Description
Each workflow process has a set of relevant data, or process specific parameters, which can be read or changed from within a workflow script. The name and syntax of these parameters, or relevant data items, are defined in the workflow designer, and are typically specific to the workflow process purpose. For example, when adding a user, an object that holds all the attributes of the new user can be a relevant data item. However, when deleting a user, the only needed relevant data item can be the distinguished name of the user to delete.

Each relevant data item will be represented in the workflow script as a variable with the same relevant data ID as defined in the workflow designer.

**ProcessData.get()**

The method changes the subject data for the current process.

**Availability**
IBM Tivoli Identity Manager 4.x

**Synopsis**

`processData.get()`

**Returns**
Returns a JavaScript object that represents the value of the relevant data item.

**Description**
This method returns a JavaScript object that represents the value of the relevant data item. There is a variable present for each relevant data item in the context of script. For performance reasons, the values are not retrieved from the workflow engine until the script specifically requests the values with this call. The returned JavaScript object is in the same syntax as defined in the workflow designer.

**Usage**
`dn = subjectDN.get();`

**ProcessData.set()**

The method changes the value of the relevant data item.

**Availability**
IBM Tivoli Identity Manager 4.x

**Synopsis**

`processData.set(value)`

**Arguments**

`value` Value to use to update the relevant data item.

**Description**
This method changes the value of the relevant data item. It not only updates the relevant data item in the script, but also in the workflow engine. The new value is a parameter to the function. The new value must be compatible with the syntax of the relevant data item as defined in the workflow designer. For example, if the relevant data item is an integer, the value cat would not be a valid parameter to this function.

**Usage**
`processData.set("engineering");`
RecertificationWorkflow

Provides extended capabilities to user recertification workflows, including audit support for the reporting and view requests functions.

Availability
IBM Tivoli Identity Manager 5.1.

Methods

\texttt{auditCompletion(person, recertPolicy, approvalDoc)}

Performs a full audit of the decisions made during a user recertification packaged approval activity, including data for reporting and view requests.

\texttt{auditTimeout(person, recertPolicy, approvalDoc)}

Perform full audit of the decisions set during a user recertification packaged approval activity timeout, including data for reporting and view requests.

\texttt{auditCompletion(person, recertPolicy, approvalDoc, auditForReports, auditForViewRequests)}

Performs an audit of the decisions made during a user recertification packaged approval activity. The value of the Boolean flag \texttt{auditForReports} determines whether an audit entry is created for reporting. The value of the Boolean flag \texttt{auditForViewRequests} determines whether an audit entry is created for view requests.

\texttt{RecertificationWorkflow.auditTimeout(person, recertPolicy, approvalDoc, auditForReports, auditForViewRequests)}

Performs an audit of the decisions set during a user recertification packaged approval activity timeout. The value of the Boolean flag \texttt{auditForReports} determines whether an audit entry is created for reporting. The value of the Boolean flag \texttt{auditForViewRequests} determines whether an audit entry is created for view requests.

Usage
\begin{verbatim}
RecertificationWorkflow.auditCompletion(Entity.get(), Policy.get(), ApprovalDocument.get())
RecertificationWorkflow.auditTimeout(Entity.get(), Policy.get(), ApprovalDocument.get(), false, true)
\end{verbatim}

Reminder

An activity to-do item reminder informs the participant that the IBM Security Identity Manager requires user action.

Availability
IBM Tivoli Identity Manager 4.x

Reminder context

Provided by
com.ibm.itim.script.extensions.ReminderExtension

Methods

\texttt{Reminder.getOriginalSubject()}

This method returns the subject of the original notification sent when the work item was first assigned.
Reminder.getXhtmlBody()
This method returns the XHTML body of the original notification sent when the work item was first assigned.

Reminder.getTextBody()
This method returns the text body of the original notification sent when the work item was first assigned.

Reminder.getRemindersSent()
This method returns the number of reminders previously sent.

Reminder.getEscalationTime()
This method returns a string that contains the date and time when the work item is escalated unless acted upon.

Reminder.getEscalationDate()
This method returns a Date containing the date and time when the work item is escalated unless acted upon.

Description
An activity to-do item reminder informs the participant that IBM Security Identity Manager requires user action.

Role
The object represents the role associated with a provisioning operation.

Availability
IBM Tivoli Identity Manager 4.x

Provided by
com.ibm.itim.script.extensions.model.RoleModelExtension

Constructor
new Role(dn)

Returns
A new Role object that represents the Role with the given DN.

Methods

getAssignmentAttributes()
Returns an array of assignment attribute names. Returns an empty array if no assignment attribute exists.

getAllAssignmentAttributes()
Returns an array of RoleAssignmentAttribute objects containing assignment attribute name, role name, and role DN. Returns an empty array if no assignment attribute exists. Returns the role assignment attributes of the whole role hierarchy.

goOwner()
Returns an array of DirectoryObjects that represent any Person that has an Owner relationship with this role.

setAssignmentAttributes()
Sets role assignment attributes of the role.

Inherits from
DirectoryObject

Synopsis
role.dn;
**Description**

The role object is available in the context of a provisioning policy.

**Note:** For more information on role assignment attributes, see [Defining assignment attributes when creating a role](#).

**Role.getAssignmentAttributes()**

The method returns an array of assignment attribute names. Returns an empty array if no assignment attribute exists.

**Availability**

IBM Security Identity Manager 6.0.

**Synopsis**

Role.getAssignmentAttributes()

**Arguments**

None

**Description**

This method is defined on the Role object and returns an array of assignment attribute names. The method returns an empty array if no assignment attribute exists.

**Usage**

```javascript
var role = new Role(roleDN);

//get assignment attributes of the role
var attributeList = role.getAssignmentAttributes();
if (attributeList.length == 0) {
    Enrole.log("script", "No assignment attribute for this role: " + role.name);
    return;
}

// print out role assignment attribute name.
for (var i=0; i < attributeList.length; i++) {
    var attrName = attributeList[i];
    Enrole.log("script","attribute name-----: " + attrName);
}
```

**Role.getAllAssignmentAttributes()**

The method returns an array of RoleAssignmentAttribute objects that contain the assignment attribute name, role name, and role DN. Returns an empty array if no assignment attribute exists. Returns the role assignment attributes of the whole role hierarchy.

**Availability**

IBM Security Identity Manager 6.0.

**Synopsis**

Role.getAllAssignmentAttributes()

**Arguments**

None

**Description**

This method is defined on the Role object and returns an array of RoleAssignmentAttribute objects. The array contains the assignment of the attribute name, role name, and role DN of the role. The method returns an empty array if no assignment attribute exists. It returns the role assignment attributes of the whole role hierarchy.
Usage

```javascript
var role = new Role(roleDN);
//get assignment attributes of the role
var attributeList = role.getAllAssignmentAttributes();
if (attributeList.length == 0) {
    Enrole.log("script", "No assignment attribute for this role: " + role.name);
    return;
}

// print out role assignment attribute name.
for (var i=0; i < attributeList.length; i++) {
    var roleAtr = attributeList[i];
    Enrole.log("script","attribute name-----: "+ roleAtr.getName());
}
```

### Role.getOwner()

The method returns an array of DirectoryObjects that represents any Person that has an Owner relationship with this role.

**Availability**

IBM Tivoli Identity Manager 5.0

**Synopsis**

Role.getOwner()

**Returns**

Array of DirectoryObjects that represents the owners of this Role or null if there are no owners.

**Usage**

```javascript
var owners = role.getOwner();
```

### Role.setAssignmentAttributes()

The method sets role assignment attributes of the role.

**Availability**

IBM Security Identity Manager 6.0.

**Synopsis**

Role.setAssignmentAttributes(String[] attributeNames)

**Arguments**

attributeNames

The array of assignment attribute names of the role. If an empty array is specified, all assignment attributes for the role are removed.

**Description**

This method is defined on the Role object and sets the role assignment attributes for a role.

**Usage**

```javascript
var roleDN = roles[0];
var role = new Role(roleDN);
var roleAtr = new Array();
roleAtr[0] = "creditlimit";
//set assignment attribute names
role.setAssignmentAttributes(roleAtr);
```

### RoleAssignmentAttribute

The object represents the role assignment attribute associated with a role.
Availability
IBM Security Identity Manager 6.0.

Methods

getName()
Returns the attribute name associated with the role assignment attribute object.

getRoleName()
Returns the name of the role. Returns an empty string if there is no name associated with the role assignment attribute object.

getRoleDN()
Returns the DN of the role. Returns an empty string if there is no DN associated with the role assignment attribute object.

Description
The RoleAssignmentAttribute object associated with the role assignment attribute.

RoleAssignmentAttribute.getName()
The method returns the name of the assignment attribute.

Availability
IBM Security Identity Manager 6.0.

Synopsis
RoleAssignmentAttribute.getName()

Arguments
None

Returns
The name of the assignment attribute.

Description
Returns the name of the assignment attribute that is defined on the role.

Usage
var role = new Role(roleDN);
//get assignment attributes of the role
var attributeList = role.getAllAssignmentAttributes();
if (attributeList.length == 0) {
    Enrole.log("script", "No assignment attribute for this role: "+ role.name);
    return;
}
// print out role assignment attribute name.
for (var i=0; i < attributeList.length; i++) {
    var roleAtr = attributeList[i];
    Enrole.log("script","attribute name-----: "+ roleAtr.getName());
}

RoleAssignmentAttribute.getRoleName()
The method returns the name of the role that has the assignment attribute defined.

Availability
IBM Security Identity Manager 6.0.

Synopsis
RoleAssignmentAttribute.getRoleName()
Arguments
None

Returns
The name of the role that has the assignment attribute defined.

Description
Returns the name of the role that has the assignment attribute defined.

Usage
var role = new Role(roleDN);
    //get assignment attributes of the role
    var attributeList = role.getAllAssignmentAttributes();
    if (attributeList.length == 0) {
        Enrole.log("script", "No assignment attribute for this role: "
            + role.name);
        return;
    }

    // print out all role names.
    for (var i=0; i < attributeList.length; i++) {
        var roleAtr = attributeList[i];
        Enrole.log("script","role name-----: " + roleAtr.getRoleName());
    }

RoleAssignmentAttribute.getRoleDN
The method returns the distinguished name of the role that defines the assignment attributes.

Availability
IBM Security Identity Manager 6.0.

Synopsis
RoleAssignmentAttribute.getRoleDN()

Arguments
None

Returns
The distinguished name of the role that defines the assignment attributes.

Description
Returns the distinguished name of the role that defines the assignment attributes.

Usage
var role = new Role(roleDN);
    //get assignment attributes of the role
    var attributeList = role.getAllAssignmentAttributes();
    if (attributeList.length == 0) {
        Enrole.log("script", "No assignment attribute for this role: "
            + role.name);
        return;
    }

    // print out the distinguished name of the role that defines
    // assignment attributes.
    for (var i=0; i < attributeList.length; i++) {
        var roleAtr = attributeList[i];
        Enrole.log("script","define role DN-----: " + roleAtr.getRoleDN());
    }
RoleAssignmentObject

The RoleAssignmentObject class is a DataObject class for role assignment data.

This class holds the assignment data that are associated with the defined role and the assigned role. The defined role is the role that holds a list of assignment attributes. The assigned role is the role to which the person is assigned.

Availability
IBM Security Identity Manager 6.0

Provided by
com.ibm.itim.script.extensions.model.RAObjectModelExtension

Constructors

new RoleAssignmentObject(RoleAssignmentObject assignmentObject)
Arguments:

assignmentObject
RoleAssignmentObject that is wrapped inside the RoleAssignmentObject.

new RoleAssignmentObject(String assignedRoleDN, String definedRoleDN)
Arguments:

assignedRoleDN
The String format of the distinguished name for the assigned role.

definedRoleDN
The String format of the distinguished name for the defined role.

Methods

addProperty()
Adds the values for specified assignment attribute.

getAssignedRoleDN()
Returns the distinguished name string for the role to which the person is assigned.

getDefinedRoleDN()
Returns the distinguished name string for the role in which the assignment attribute is defined.

getChanges()
Returns the changes made to this RoleAssignmentObject.

getProperty()
Returns the values of the property specified by the assignment attribute name.

getPropertyNames()
Returns a list of role assignment attribute names.

removeProperty()
Removes the values for the specified assignment attribute name.

setProperty()
Sets the values for a specified assignment attribute.
Description
RoleAssignmentObject contains the role assignment data, including the assigned role DN, the defined role DN and attribute values.

RoleAssignmentObject.getAssignedRoleDN()

The method returns the distinguished name string for the role to which a person is assigned.

Availability
IBM Security Identity Manager 6.0.

Synopsis
roleAssignmentObject.getAssignedRoleDN()

Arguments
None

Returns
The distinguished name string for the role to which a person is assigned.

Usage
definedRoleDN = "globalid=111";
definedRoleDN = "globalid=222";
var assignmentObj = new RoleAssignmentObject(assignedRoleDN, definedRoleDN);

var assignedRoleDN2 = assignmentObj.getAssignedRoleDN();

RoleAssignmentObject.getDefinedRoleDN()

The method returns the distinguished name string for the role in which the assignment attribute is defined.

Availability
IBM Security Identity Manager 6.0

Synopsis
roleAssignmentObject.getDefinedRoleDN()

Arguments
None

Returns
Returns the distinguished name string for the role in which the assignment attribute is defined.

Description
This method returns the distinguished name string for the role to which the person is assigned.

Usage
definedRoleDN = "globalid=111";
definedRoleDN = "globalid=222";
var assignmentObj = new RoleAssignmentObject(assignedRoleDN, definedRoleDN);

var definedRoleDN2 = assignmentObj.getDefinedRoleDN();

RoleAssignmentObject.addProperty()

Use this method to add the values for specified assignment attribute.
Availability
IBM Security Identity Manager 6.0.

Synopsis
RoleAssignmentObject.addProperty(name, value)

Arguments
name String representing the name of the assignment attribute to be added.
value The value to be added.

Description
This method changes the value of the specified assignment attribute or adds the specified assignment attribute if it does not exist. This change is made locally to the script environment, not to the data store.

Usage
// Create assignment object with assigned role dn and defined role dn.
var assignmentObj = new RoleAssignmentObject("eruid=1111,dc=com", "eruid=2222,dc=com");
// Add some assignment attribute with values.
assignmentObj.addProperty("attr1", ["attr1val1","attr2val1"]);
assignmentObj.addProperty("attr2", ["attr2val1"]);
assignmentObj.addProperty("attr2", ["attr2val2"]);

RoleAssignmentObject.getChanges()
The method returns the changes made to an entity.

Availability
IBM Security Identity Manager 6.0.

Synopsis
RoleAssignmentObject.getChanges()

Returns
An array of change objects. If there are no changes, an empty array is returned. Each element in the array is an AttributeChangeOperation.

Description
This method returns the changes made to the entity. These changes are represented by change objects with the following members:

attr String name of the attribute that is being changed.
op An integer that identifies the type of change that is being made. The enumerated values are 1 for add, 2 for replace, and 3 for remove.
values An array of objects that can be either added, removed, or replaced.

The changes are returned as an array of these change objects. If there are no changes, an empty array is returned.

Usage
changes = assignmentObj.getChanges();
for (i = 0; i < changes.length; i++) {
    name = changes[i].attr;
    if (changes[i].op == 1) {
        ...
    } else if (changes[i].op == 2) {
        ...
    } else if (changes[i].op == 3) {
        ...
    }
}
RoleAssignmentObject.getProperty()

The method returns the values of the assignment attribute specified by the given name.

Availability
IBM Security Identity Manager 6.0.

Synopsis
RoleAssignmentObject.getProperty(name)

Arguments
name  String representing the name of the assignment attribute to return.

Returns
The array of strings that represents the values for an assignment attribute. If the specified assignment attribute does not exist, an empty array is returned.

Description
This method returns the values of the assignment attribute specified by the given name. If the specified assignment attribute does not exist, an empty array is returned.

Usage
// create assignment object with assigned role dn and defined role dn.
var assignmentObj = new RoleAssignmentObject("eruid=1111,dc=com",
"eruid=2222,dc=com");
assignmentObj.addProperty("attr1", ["attr1val1", "attr1val2"]);

// get assignment attribute values for attr1.
var attrValues = assignmentObj.getProperty("attr1");
var attrValuesStr = "";
for (var j=0; j<attrValues.length; j++) {
    attrValuesStr += attrValues[j] + ", ";
Enrole.log("script", "The values for attr1:" + attrValuesStr);

RoleAssignmentObject.getPropertyNames()

The method returns a list of assignment attributes.

Availability
IBM Security Identity Manager 6.0.

Synopsis
RoleAssignmentObject.getPropertyNames()

Returns
An array of strings.

Description
This method returns a list of assignment attributes as an array of strings.

Usage
properties = RoleAssignmentObject.getPropertyNames();

RoleAssignmentObject.removeProperty()

The method removes the assignment attribute specified by the given name.
Availability
IBM Security Identity Manager 6.0.

Synopsis
RoleAssignmentObject.removeProperty(name)

Arguments
name String representing the name of the assignment attribute to remove.

Description
This method removes the specified assignment attribute. This change is made locally to the script environment, not to the data store.

Usage RoleAssignmentObject.removeProperty("assignmentAttr1");

RoleAssignmentObject.setProperty()

The method sets the value of the specified assignment attribute.

Availability
IBM Security Identity Manager 6.0.

Synopsis
RoleAssignmentObject.setProperty(name, value)

Arguments
name String representing the name of the assignment attribute to be created or modified.

type value Specifies the value to which the assignment attribute is set.

Description
This method changes the value of the specified assignment attribute, or adds the specified assignment attribute if it does not exist. This change is made locally to the script environment, not to the data store.

Usage RoleAssignmentObject.setProperty("attr1",["val1","val2"]);

RoleSearch

The object searches for a role.

Availability
IBM Tivoli Identity Manager 4.x

Provided by
com.ibm.itim.script.extensions.model.RoleModelExtension

Constructor
new RoleSearch()

Returns
The newly created and initialized role search object.

Methods

searchByName()
Search for a role by name.

searchByURI()
Search for a role by URI within an organizational container.
RoleSearch.searchByName()

The method searches for a role by a name.

Availability
   IBM Tivoli Identity Manager 4.6

Synopsis
   RoleSearch.searchByName(name)

Arguments
   name   The role name to use as the basis for the search.

Returns
   Array of DirectoryObjects that represents a role.

Description
   Given the name of a role, locate the Role entity. Will return null if there is not exactly one matching role.

Usage
   // Given the name of a role, see if it exists and log its
   // description
   var roles = (new RoleSearch()).searchByName("testRole");
   if (roles.length >= 1) {
     if (roles[0].getProperty("errolename"))[0] == "testRole") {
       Enrole.log("script", "The Role " + roles[0].getProperty("errolename"))[0] +
       "has Description :" + roles[0].getProperty("description")[0];
     }
   }

RoleSearch.searchByURI()

The method finds a role by URI in an organizational container.

Availability
   IBM Security Identity Manager 6.0.

Synopsis
   RoleSearch.searchByURI(containerDN, uri)

Arguments
   Container DN
      String representing the distinguished name of the organizational container.
   uri   String representing the URI of the role.

Returns
   A Role object

Description
   Given the distinguished name of the organizational container and the role URI, this method finds the container. If the role is not found, this function returns null. If more than one role is found, this function throws a scripting exception.

Usage
   var role = (new RoleSearch()).searchByURI(container.dn, uri);
   if (role != null) {
     Enrole.log("script", "Found " + role.getProperty("errolename") );}
SeparationOfDutyRuleViolation

Object that provides information about a specific separation of duty rule violation. Use this object to get specific information about a separation of duty policy violation. This object cannot be created for use by the user. The user can work only with SeparationOfDutyRuleViolation objects that the system has generated as part of the approveSoDViolation workflow.

Availability
IBM Tivoli Identity Manager 5.1.

Provided by
com.ibm.itim.script.wrappers.generic.IRuleResultWrapper

Methods

getName()
Returns the name of the separation of duty policy rule to which this violation corresponds.

getViolationString()
Provides a string that represents the list of roles in violation. It describes the roles the person has that are in violation and which role in a separation of duty rule they correspond to. The role lists might be different due to role hierarchy.

getViolationStringHTMLTable()
Returns a string version of the roles in violation for use in an HTML table or email template.

getPolicyName()
Returns the name of the separation of duty policy which contains the rule in violation.

getPolicyDescription()
Returns a description of the separation of duty policy.

getPolicyOwnerDNs()
Returns a collection of the distinguished names of one or more separation of duty policy owners.

getCardinality()
Returns string that represents the number of allowed roles in the separation of duty policy rule in violation.

Service

The object represents the service associated with a provisioning operation.

Availability
IBM Tivoli Identity Manager 4.x

Provided by
com.ibm.itim.script.extensions.model.ServiceModelExtension

Constructor
new Service(dn)

Returns
A new Service object that represents the Service with the DN.
Inherits From
  DirectoryObject

Synopsis
  service.dn;

Description
  The service object is available in the context of a Provisioning Policy and
  Service Selection Policy.

ServiceSearch

Use the object to provide searching capability for IBM Security Identity Manager
services.

Availability
  IBM Tivoli Identity Manager 4.x
  Provisioning Policy context
  Service Selection Policy context

Provided by
  com.ibm.itim.script.extensions.model.ServiceModelExtension

Methods
  searchByFilter()
    Search for a service by a filter.
  searchByName()
    Search for a service by a name.
  searchByURI()
    Search for a service by URI in an organizational container.
  searchForClosestToPerson()
    Search for the closest Service to a person.

Description
  This object is used to provide searching capability for IBM Security
  Identity Manager services.

ServiceSearch.searchByFilter()

The method searches for a service by a filter.

Availability
  IBM Tivoli Identity Manager 4.x

Synopsis
  ServiceSearch.searchByFilter(filter, scope)

Arguments
  filter    LDAP search filter that defines the criteria for returned
            containers to meet. The filter must be in the format defined
            by RFC2254.
  scope     Optional search scope. Use 1 for One Level Scope and 2 for
            SubTree Scope. One Level Scope is the default scope.

Returns
  An array of DirectoryObjects representing the results of the
  search.
Description
This method searches for a service by a filter.

Usage
searchResult1 = ServiceSearch.searchByFilter("(erntlocalservername=*srv)", 2);

// use default one level scope, put statement on one line
searchResult2 = ServiceSearch.searchByFilter("(erntlocalservername=*srv)");

ServiceSearch.searchByName()
The method searches for a service by name.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis
ServiceSearch.searchByName(name, profileName, scope)

Arguments
name The service name, provided as a string, to use as the basis for the search.

profileName Optional profile name, provided as a string. The profile name of the service to use as the basis for the search.

scope Optional search scope, provided as an int. Use 1 for One Level Scope and 2 for Scope. One Level Scope is the default scope. When you use this method in workflow JavaScripts, set the scope parameter to SubTree because the logical search context is limited to the tenant above the default organization. In this context, setting the scope to One Level Scope returns empty results during a search because there are no services at the tenant level.

Returns
An array of DirectoryObjects representing the results of the search.

Description
This method searches for a service by a name.

Usage
searchResult1 = ServiceSearch.searchByName("US NT Service", 2);

// use default one level scope
searchResult2 = ServiceSearch.searchByName("US NT Service");

ServiceSearch.searchByURI()
The method finds a service by URI in an organizational container.

Availability
IBM Security Identity Manager 6.0.

Synopsis
ServiceSearch.searchByURI(containerDN, uri)

Arguments
Container DN
String representing the distinguished name of the organizational container.

uri String representing the URI of the service.

Returns
A Service object

Description
Given the distinguished name of the organizational container and the service URI, this method finds the service. If the service is not found, this function returns null. If more than one service is found, this function throws a scripting exception.

Usage
var service = (new ServiceSearch()).searchByURI(container.dn, uri);
if (service != null) {
Enrole.log("script", "Found " + service.getProperty("erservicename") );
}

ServiceSearch.searchForClosestToPerson()
The method searches for a service closest to a person.

Availability
IBM Tivoli Identity Manager 4.x

Synopsis
ServiceSearch.searchForClosestToPerson(person, profileName)

Arguments

person The DirectoryObject representing a person to use as the basis for the search.

profileName Optional service profile name.

Returns
An array of DirectoryObjects representing the results of the search.

Description
This method searches for a service closest to a person.

Usage
//Search for AIX service closest to the person.
searchResult1 = ServiceSearch.searchForClosestToPerson(subject, "PosixAixProfile");

//Search for any service closest to the person.
searchResult2 = ServiceSearch.searchForClosestToPerson(subject);
Chapter 11. Provisioning policy parameter usage scenarios

The following scenarios illustrate usage of provisioning policy parameters.

**Scenario 1: No attributes defined**

If no parameter values are selected for a multi-valued attribute, all values are valid for this attribute.

If a parameter is added for a multi-valued attribute with the parameter type as `Allowed (valid)`, all other values for this attribute are implicitly excluded for this policy.

If an attribute value is added to a policy as valid, all other values are implicitly excluded for that parameter for the policy.

For multiple applicable entitlements, a valid attribute value is determined by the join directive for the attribute. See the following scenarios.

**Scenario 2: Priority-based provisioning policy join directive**

The following table identifies two examples of provisioning policies:

<table>
<thead>
<tr>
<th>Policy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy 1</td>
<td>Priority = 1 Attribute: <code>erdivision</code> = divisionA, <code>enforcement</code> = DEFAULT</td>
</tr>
<tr>
<td>Policy 2</td>
<td>Priority = 2 Attribute: <code>erdivision</code> = divisionB, <code>enforcement</code> = MANDATORY</td>
</tr>
</tbody>
</table>

Because Policy 1 has a higher priority, only Policy 1’s definition for the `erdivision` attribute is used. Policy 2’s definition for the `erdivision` attribute is ignored.

During policy validation, including reconciliation with policy check option turned on, `divisionA` might exist on the `erdivision` attribute. All other values are valid, because the only policy that is being considered in a priority join is Policy 1, which has DEFAULT enforcement on `erdivision`. DEFAULT enforcement by itself is interpreted as valid for all values, but the default value is the value specified on the new account.

**Note:** A priority join directive is the default join directive for all single-valued and string-typed attributes.

**Scenario 3: Union-based provisioning policy join directive**

The following table identifies two example provisioning policies:

<table>
<thead>
<tr>
<th>Policy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy 1</td>
<td>Priority = 1 Attribute: <code>localgroup</code> = groupA, <code>enforcement</code> = DEFAULT</td>
</tr>
<tr>
<td>Policy 2</td>
<td>Priority = 2 Attribute: <code>localgroup</code> = groupB, <code>enforcement</code> = MANDATORY</td>
</tr>
</tbody>
</table>
Because the join directive is defined as UNION, the resulting policy uses the following definitions for the policies:

- During account creation, localgroup is defined as groupA and groupB.
- During reconciliations, localgroup is defined as groupB if the attribute is undefined or incorrectly defined.

**Note:** A union join directive is the default join directive for multi-valued attributes.
Chapter 12. Provisioning policy entitlement parameters

Provisioning policy parameters help system administrators define the attribute values that are required and the values that are allowed.

The following parameter types are valid:
- Constant value
- Null
- JavaScript
- Regular expression

The provisioning parameters in an entitlement can be statically or dynamically defined. Parameters are defined statically by selecting the constant parameter type and specifying literal values, such as strings or integers. For example, an attribute can be defined as Domain Users or Power® Users. A dynamically defined parameter value can be based on a JavaScript function. A range of values can be defined that use a regular expression.

Parameters can also be specified as `null`, indicating that the parameter does not have a value. This situation is equivalent to having a JavaScript parameter type with a value of `return null;`

Provisioning parameters for single-valued attributes can be based only on JavaScript code or a constant. The provisioning parameters of a multi-valued attribute can use a constant, JavaScript code, or regular expression for their values.

However, a regular expression can be used only for provisioning parameter enforcement of the Allowed or Excluded type.

**Provisioning policy constant**

A static, constant value can be assigned to an entitlement parameter for a single or multivalued attribute with the provisioning policy Constant parameter type. You can define a provisioning parameter with a literal static value. You can enter the value or select a value based on the field widget.

**Provisioning policy Null types**

The null parameter type can be used to specify a null value for an account attribute. If the value of a parameter is specified as null with mandatory enforcement, no value is valid for that attribute. You can specifically define null value for the provisioning parameter, which is equivalent to specifying empty for the value.

**Provisioning policy JavaScript functions**

You can use a script to define provisioning parameters.

The provisioning parameters of an entitlement within a provisioning policy can be defined by a script. The context of the script is:
- The person for whom the entitlement is being enforced.
- The service the entitlement is protecting.
- The eruid attribute of the target account.

The context of the script includes the following elements:

**Subject**
- Owner of the account.

**Service**
- Service on which the account exists or to be created.

**uid**
- User ID of the account.

**Context**
- Information about the parameter evaluation, which can be validation of a new account or validation of existing account.

A special object named *parameters* is available for eruid to evaluate the script in the context of provisioning policy parameters. To obtain its value, use the following syntax:

```
parameters.eruid[0]
```

The value of zero in this syntax returns the first value of the array object.

A JavaScript object named *subject* represents a user for whom the entitlement is being enforced. The service is represented by another JavaScript data model entity named *service*. The script author uses both the subject and service object to access attributes of these objects.

The values of attributes of objects that are part of the evaluation context can also be retrieved with the IBM Security Identity Manager custom JavaScript functions.

To use JavaScript to define the value of an attribute, the JavaScript parameter type must be selected. Select **JavaScript/Constant** in the **Expression Type** field.

The following examples demonstrate the use of IBM Security Identity Manager custom JavaScript functions within provisioning policies. For a complete reference to all custom JavaScript functions, see the JavaScript Extension Reference.

**Person attributes**

**Syntax:**
```
subject.getProperty(String rowAttrName)
```

**Example:**
```
subject.getProperty("sn")[0];
```

**Example:**
```
# Concatenates user's given name and family name with space in between.
# Resulting string value may be used to on account attribute such as Description.
{subject.getProperty("givenname")[0] + " " + subject.getProperty("sn")[0];}
```

**Example:**
```
# Set a user's Password attribute to the user's Shared Secret Attribute
# (if the account is automatically provisioned)
{
```
function passInit()
{
    var password = subject.getProperty("ersharedsecret");
    if (password.length > 0){
        return password[0];
    } else {
        return "";
    }
}

Search for person

Syntax:
PersonSearch.searchByFilter(String profileName, String filter, [int scope])

where scope = 1 is a single level search and scope = 2 is a subtree search.

Example:
PersonSearch.searchByFilter("Person", "(sn=Smith)", 1);

Search for service

Syntax:
ServiceSearch.searchByFilter(String filter, [int scope])

where scope = 1 is a single level search and scope = 2 is a subtree search.

Example:
ServiceSearch.searchByFilter("(erntlocalservername=*srv)", 1);

Service closest to the person

Syntax:
ServiceSearch.searchForClosestToPerson(Person person, [int scope])

where scope = 1 is a single level search and scope = 2 is a subtree search.

Example:
ServiceSearch.searchForClosestToPerson(subject);

Name of the business unit in which the person is located

Syntax:
subject.getProperty(String propertyName)

Example:
subject.getProperty("Parent")[0].name;

Specifying the current account Uid

Syntax:
uid = parameters.eruid[0];

Example:
var accountId = parameters.eruid[0];
**Enrole.toGeneralizedTime statement**

Syntax:
Enrole.toGeneralizedTime(Date date)

Examples:

Using the function to return today's date string:
var gt = Enrole.toGeneralizedTime(new Date());

Using the function to return today's date string as a default attribute:
{Enrole.toGeneralizedTime(new Date())}

**Enrole.toMilliseconds statement**

Syntax:
Enrole.toMilliseconds(String generalizedTime)

Examples:

var millis = Enrole.toMilliseconds("200101012004Z");
var date = new Date(millis);

**Provisioning policy regular expressions**

Regular expressions are used to define a matching pattern that is checked against given text. Within IBM Security Identity Manager, regular expressions define allowed and excluded parameter values.

Within IBM Security Identity Manager, regular expressions define allowed and excluded parameter values. Parameter values with regular expressions are used against existing attribute values to determine which ones are valid.

To use a regular expression for a provisioning parameter value, select **Regular Expression** in the Expression Type menu.

**Note:** Regular Expression can be used only with excluded or allowed attributes. See the regexp Java library for a syntax reference.
Chapter 13. Service selection policy JavaScript

A service selection policy identifies the service type for the service returned, and the JavaScript specifies the service. For example, the service definition can be based on attributes of an account owner.

Service selection policy JavaScript objects

The service selection policy JavaScript returns an object that represents a IBM Security Identity Manager service entity.

The “subject” JavaScript object is a Person object that represents the account owner. Attributes of the Person can be used to construct filters to search and return the service. The ServiceModelExtension is available for Service Selection policy by default.

The following list includes APIs for the ServiceSearch JavaScript object that can be used to return the service:
- ServiceSearch.searchByName
- ServiceSearch.searchByFilter
- ServiceSearch.searchForClosestToPerson

See a JavaScript API reference guide for detailed information for these APIs.

Service selection policy script example

This section includes examples of Service Selection policy scripts.

Service selection based on family name

The following script returns a service instance based on the family name of the account owner. Other person attributes such as job title and location can be used to select service, as in this example.

```javascript
function selectService() {
    var sn = subject.getProperty("sn")[0];
    var serviceInstance = null;
    if(sn=="Jones") {
        serviceInstanceArr = ServiceSearch.searchByFilter(
            "(erservicename=NT40X)", 1);
        if (serviceInstanceArr != null && serviceInstanceArr.length > 0)
            serviceInstance = serviceInstanceArr[0];
    } else {
        serviceInstanceArr = ServiceSearch.searchByFilter(
            "(erservicename=NT40Y)", 1);
        if (serviceInstanceArr != null && serviceInstanceArr.length > 0)
            serviceInstance = serviceInstanceArr[0];
    }
    return serviceInstance;
}
return selectService();
```
Searching for the closest service to the person

The following example searches for the service closest to the level of the person, based on the organization tree structure.

```javascript
function selectService() {
    var services = ServiceSearch.searchForClosestToPerson(subject);
    if (services != null && services.length > 0) {
        return services[0];
    }
}
return selectService();
```
Chapter 14. SubForm control type

The SubForm control type provides a means to use custom user interfaces for complex multi-valued attributes.

This control type is used infrequently by some IBM Security Identity Manager adapters.

SubForm is a special control type used to start a servlet, JSP, or static HTML page from a window that opens from a custom IBM Security Identity Manager form. Use subforms to submit an arbitrary number of parameter names and values to a custom servlet or JSP. They are used to create custom user interfaces for complex multi-valued attributes.

Table 11. SubForm parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>customServletURI</td>
<td>The URI to the servlet, JSP, or static HTML page to be started from the main form. If a servlet is implemented and deployed in the default web application for IBM Security Identity Manager, the value for this parameter is the same as the <em>URL-pattern</em> value defined by <em>web.xml</em> in the <em>servlet-mapping</em> tag, without the slash (/). If a JSP is implemented, the value for this parameter is the JSP file name that includes the <em>jsp</em> file extension. This parameter is required on all subforms.</td>
<td>Servlet name or JSP file name such as sample.jsp</td>
</tr>
</tbody>
</table>

For more information, see the subform example and other information in the _ISIM_HOME/extensions/examples/_ subdirectory that IBM Security Identity Manager provides.

SubForm contextual parameters

As a child element of a main form, a SubForm is passed contextual parameters that help identify the context from which it is started.

These contextual parameters are included in the HTTP Request that brings up the SubForm:
Table 12. SubForm parameters

<table>
<thead>
<tr>
<th>HTTP (contextual) Parameter Name</th>
<th>Person Create</th>
<th>Person Modify</th>
<th>Account Create</th>
<th>Account Modify</th>
</tr>
</thead>
<tbody>
<tr>
<td>target_dn</td>
<td>empty</td>
<td>DN of Person</td>
<td>DN of account owner</td>
<td>DN of the account</td>
</tr>
<tr>
<td>container_dn</td>
<td>DN of the organization tree container where the Person is created.</td>
<td>DN of the organization tree container where the person is located.</td>
<td>DN of account owner</td>
<td>DN of the account owner</td>
</tr>
<tr>
<td>search_base</td>
<td>empty</td>
<td>empty</td>
<td>DN of service</td>
<td>DN of the service instance on which an account is provisioned</td>
</tr>
</tbody>
</table>

To assign the target_dn HTTP parameter value to a String declared inside a servlet:

```java
String targetDN = request.getParameter("target_dn");
```

**Account Modify example**

For example, for Account Modify, the value of contextual parameters are:

**target_dn**
- Is the DN of the entity whose attributes are displayed on the main form. If the SubForm is placed on a RACF® account form, this parameter value is the `DistinguishedName` of the RACF account.

**container_dn**
- Is the entity container or parent. For example, if the SubForm is placed on a Person form, this parameter value is the `DistinguishedName` of the parent or container of the person. The container can be an organization, organizational unit, admin domain, or location.

**search_base**
- For example, if the SubForm is placed on a RACF account form, this parameter value is the `DistinguishedName` for the RACF service instance on which the account is provisioned.

**HTTP request parameter naming convention**

A naming convention used on SubForm parameters prevents collisions with other parameters (such as contextual parameters).

The naming convention for SubForm parameters is:
```
[prefix].[attributename].[parametername]
```

where:

- **prefix** property.data

- **attributename**
  - Name of the attribute on which the SubForm is placed on the main form.
**parametername**
Name used in the SubForm Editor dialog. For example, an HTTP parameter named `property.erracfconnectgroup.objectClass` would contain the value defined in the SubForm editor dialog assigned to `objectClass`.

To assign this value to a string declared inside a servlet:
```java
String objectClass = request.getParameter("property.data.erracfconnectgroup.objectClass");
```

---

**Process to write a SubForm**

To write a custom SubForm, create a servlet that generates the HTML user interface to manage the value of the attribute.

To save the value, create an instance of `com.ibm.itim.common.AttributeValue` and bind it to a user's `HttpSession` with the key defined in `com.ibm.itim.webclient.util.FormData` (on one line):

```java
AttributeValue av = new AttributeValue("attributename", "customValue");
HttpSession session = request.getSession(false);
    session.setAttribute("subFormAttrValue", av);
```

This ensures that the value gets picked up and added to the form data collected from the fields when the main form is submitted.
Chapter 15. Supplemental property files

The following section provides detailed information about the property keys and values contained in the IBM Security Identity Manager supplemental property files.

Properties files

Java properties files define attributes that allow customizing and control of the Java software.

Standard system properties files and custom properties files are used to configure user preferences and user customization. A Java properties file defines the values of named resources that can specify program options such as database access information, environment settings, and special features and functions.

A properties file defines named resources with a property key and value pair format:

property-key-name=value

The property-key-name is an identifier for the resource. The value is usually the name of the actual Java object that provides the resource, or a String representing the value of the property key, such as database.name=itimdb. The statement syntax allows spaces before and after the equal (=) sign, and can span multiple lines if you place a line continuation character \ (a backslash) at the end of the line. For more information about statement syntax, see Java language references.

IBM Security Identity Manager uses a number of properties files to control the program and to allow user customization of special features.

Modifiable property files

This table lists the IBM Security Identity Manager properties files that you can modify.

Table 13 lists the IBM Security Identity Manager properties files. Most files are in the ISIM_HOME\data\ directory.

Additional properties files are not configurable. Do not modify them.

<table>
<thead>
<tr>
<th>Property file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adhocreporting</td>
<td>The adhocreporting.properties file supports the custom reporting module.</td>
</tr>
<tr>
<td>CustomLabels</td>
<td>The property key and value pairs in the CustomLabels.properties file are used by the IBM Security Identity Manager user interface to display the label text for forms.</td>
</tr>
<tr>
<td>DataBaseFunctions.conf</td>
<td>The custom reporting feature of IBM Security Identity Manager allows you to use database functions when designing custom report templates.</td>
</tr>
</tbody>
</table>
Table 13. Properties files (continued)

<table>
<thead>
<tr>
<th>Property file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enRole</td>
<td>The enRole.properties system configuration file contains many of the properties used to configure IBM Security Identity Manager.</td>
</tr>
<tr>
<td>enroleAuditing</td>
<td>The property key and value pairs in the enroleAuditing.properties file are used to enable or disable the tracking of changes made by a IBM Security Identity Manager user to business objects such as person, location, service, and other objects, or configuration of the system.</td>
</tr>
<tr>
<td>enRoleAuthentication</td>
<td>The enRoleAuthentication.properties file specifies the type of method that is used by the Security Identity Manager Server to authenticate users and identifies the Java object that provides the specified authentication mechanism.</td>
</tr>
<tr>
<td>enRoleDatabase</td>
<td>The enRoleDatabase.properties file specifies attributes that support the relational database used by IBM Security Identity Manager.</td>
</tr>
<tr>
<td>enRoleLDAPConnection</td>
<td>The enRoleLDAPConnections.properties file provides standard configuration settings that allow successful communication between IBM Security Identity Manager and the LDAP directory server.</td>
</tr>
<tr>
<td>enRoleLogging</td>
<td>The enRoleLogging.properties file specifies attributes that govern the operation of the jlog logging and tracing API that is bundled with IBM Security Identity Manager.</td>
</tr>
<tr>
<td>enRoleMail</td>
<td>The enRoleMail.properties file contains attributes that specify the mail transport protocol that is used by the JavaMail API and other IBM Security Identity Manager application-specific properties. You must provide the values for the application-specific properties.</td>
</tr>
<tr>
<td>enrolepolicies</td>
<td>The enrolepolicies.properties file provides standard and custom settings that support the functions of the provisioning policy.</td>
</tr>
<tr>
<td>enroleworkflow</td>
<td>The enroleworkflow.properties file specifies the XML file mappings for system-defined workflows.</td>
</tr>
<tr>
<td>fesiextensions</td>
<td>The fesiextensions.properties file (deprecated) provides support for Free EcmaScript Interpreter (FESI) JavaScript extensions before Version 5.0 of IBM Security Identity Manager. Do not author new extensions using this deprecated architecture.</td>
</tr>
<tr>
<td>helpmappings</td>
<td>The helpmappings.properties file allows a customer to replace the installed IBM Security Identity Manager help system with an alternative help system.</td>
</tr>
<tr>
<td>reportingLabels</td>
<td>This properties file is like other labels-related properties files such as labels.properties, or customLabels.properties, and holds labels that are used by Reports.</td>
</tr>
<tr>
<td>reporttabledeny</td>
<td>By default, this property holds a list of IBM Security Identity Manager tables that are used by various IBM Security Identity Manager components to store internal or configuration data that is inappropriate for a report.</td>
</tr>
<tr>
<td>scriptframework</td>
<td>For all new JavaScript extensions, use the scriptframework.properties file to configure script extensions and other scripting functions.</td>
</tr>
</tbody>
</table>
Table 13. Properties files (continued)

<table>
<thead>
<tr>
<th>Property file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SelfServiceHelp</td>
<td>The SelfServiceHelp.properties file can be used to redirect help to a custom location for customers who want to have their own help content for the self-service user interface.</td>
</tr>
<tr>
<td>SelfServiceHomePage</td>
<td>The SelfServiceHomePage.properties file is used to configure the sections of the initially installed home page for the self-service user interface. You can add or remove tasks, and update icon URLs and labels of the home page from this file.</td>
</tr>
<tr>
<td>SelfServiceScreenText</td>
<td>The SelfServiceScreenText.properties file is a resource bundle containing the labels for the self-service user interface.</td>
</tr>
<tr>
<td>SelfServiceUI</td>
<td>The SelfServiceUI.properties file controls miscellaneous properties of the self-service user interface.</td>
</tr>
<tr>
<td>ui</td>
<td>The ui.properties file specifies attributes that affect the operation and display of the IBM Security Identity Manager graphical user interface.</td>
</tr>
</tbody>
</table>

Non-modifiable properties files

Some property files are not configurable. Do not modify them.

Table 14 lists the remaining property files that are used by IBM Security Identity Manager. In all cases, these files are not configurable. Do not modify them.

Table 14. Non-modifiable properties files

<table>
<thead>
<tr>
<th>Property file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfigErrorMessages</td>
<td>This file is used by the <code>runConfig</code> utility and contains configuration error messages in English. Do not modify.</td>
</tr>
<tr>
<td>ConfigLabels</td>
<td>This file is used by the <code>runConfig</code> utility and contains IBM Security Identity Manager Console display labels in English. Do not modify.</td>
</tr>
<tr>
<td>ConfigMessages</td>
<td>This file is used by the <code>runConfig</code> utility and contains configuration instructions and normal messages in English. Do not modify.</td>
</tr>
<tr>
<td>CustomForms</td>
<td>This file is used for form generation, form display, and form design. Do not modify.</td>
</tr>
<tr>
<td>CustomThemes</td>
<td>Do not modify. This file has custom themes used by applets to support accessibility.</td>
</tr>
<tr>
<td>dataSynchronization</td>
<td>This file is used by the IBM Security Identity Manager Data Services component to define data replication for runtime execution optimization purpose. Do not modify.</td>
</tr>
<tr>
<td>Dsm12RootDSE</td>
<td>This file is used for searching a root DSE (LDAP) to return a collection of attributes about the IBM Security Identity Manager server. Do not modify.</td>
</tr>
<tr>
<td>Dsm12Schema</td>
<td>This file is used for searching a schema (LDAP) to return object classes and attributes specified in this file. Do not modify.</td>
</tr>
<tr>
<td>Property file name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>encryptionKey</td>
<td>This file is used to store the encryption password information in the IBM Security Identity Manager Console. Do not modify.</td>
</tr>
<tr>
<td>enRole2Ldif</td>
<td>This file is now deprecated and was used for migration from enRole 3.x to 4.4. Do not modify.</td>
</tr>
<tr>
<td>enRoleEntityHiddenAttributes</td>
<td>This file is used to filter out LDAP attributes for each entity type available for mapping. For example, Organization, BPPOrganization, Person, BPPerson. Do not modify.</td>
</tr>
<tr>
<td>enRoleFonts</td>
<td>This file specifies font names for locale languages. Do not modify.</td>
</tr>
<tr>
<td>enRoleHelp</td>
<td>This file contains a list of operations that are not in the workflow designer. Do not modify.</td>
</tr>
<tr>
<td>enRoleHiddenAttributes</td>
<td>This file contains the attributes of each object class (for example, person, service, account, organization unit) that are invisible to the IBM Security Identity Manager Console. This hidden attribute list contains mostly attributes used by the system. Do not modify.</td>
</tr>
<tr>
<td>enRoleHiddenOperations</td>
<td>Do not modify.</td>
</tr>
<tr>
<td>enRoleHiddenSearchAttributes</td>
<td>Attributes listed in this file are not in search activities or in any pending and completed request details. This file is used to filter out process data attributes that must not be displayed in the user interface. Do not remove the existing entries in this file, otherwise the search function on these attributes fails. Do not modify.</td>
</tr>
<tr>
<td>enroleStartup</td>
<td>This file is used to specify startup activities in the WebSphere Application Server environment. Do not modify.</td>
</tr>
<tr>
<td>enRoleUnchangedAttributes</td>
<td>This file is used by the directory server upgrade utility. Do not modify.</td>
</tr>
<tr>
<td>enRoleValidateAttributes</td>
<td>This file is used internally by the IBM Tivoli Identity Manager Express Server for entity schema attribute mapping. Do not modify.</td>
</tr>
<tr>
<td>entitlementHiddenAttributes</td>
<td>This file is used by the Tivoli Identity Manager Express Server for filtering out system managed attributes from displaying on the available entitlement parameter selection. Do not modify.</td>
</tr>
<tr>
<td>expressHiddenAttributes</td>
<td>This file was used by the Tivoli Identity Manager Express Server at Version 4.6.x. Do not modify.</td>
</tr>
<tr>
<td>HighContrastBigFontTheme</td>
<td>This file is used to specify the appearance of a high contrast, large font for applet accessibility. Do not modify.</td>
</tr>
<tr>
<td>HighContrastTheme</td>
<td>This file is used to specify high contrast display values for applet accessibility. Do not modify.</td>
</tr>
<tr>
<td>ibmSchemaSyntax</td>
<td>This file is used by LDAP configuration during IBM Security Identity Manager installation. Do not modify.</td>
</tr>
<tr>
<td>iplanetSchemaSyntax</td>
<td>This file is used by LDAP configuration during IBM Security Identity Manager installation. Do not modify.</td>
</tr>
</tbody>
</table>
Table 14. Non-modifiable properties files (continued)

<table>
<thead>
<tr>
<th>Property file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>itiminstaller</td>
<td>This file is during IBM Security Identity Manager installation. Do not modify.</td>
</tr>
<tr>
<td>Labels</td>
<td>This file contains English labels for the UI display. Do not modify.</td>
</tr>
<tr>
<td>Messages</td>
<td>This file contains all normal messages that IBM Security Identity Manager uses to communicate with users. Do not modify.</td>
</tr>
<tr>
<td>passwordrules</td>
<td>This file is used to specify the custom class for generating passwords. IBM Security Identity Manager provides a default password generator. In the sample passwordrules.properties, the first line contains the class name. The second line defines the input requires by the class defined in line 1. Your site might require additional rules for use in production. Do not modify.</td>
</tr>
<tr>
<td>pimDataSync</td>
<td>This file is used by the IBM Security Identity Manager Data Services component to define shared access data replication for shared access runtime execution optimization purpose. Do not modify.</td>
</tr>
<tr>
<td>pimWorkflowDataSyntax</td>
<td>This file defines workflow data syntax for the Shared Access Module. Do not modify.</td>
</tr>
<tr>
<td>pimWorkflowextensions</td>
<td>This file defines workflow extensions for Shared Access Module. Do not modify.</td>
</tr>
<tr>
<td>platformcontext</td>
<td>This file specifies provisioning platform context information. Do not modify.</td>
</tr>
<tr>
<td>Properties</td>
<td>This file is the top-level properties file that indicates the actual properties file path. Do not modify.</td>
</tr>
<tr>
<td>subform</td>
<td>This file is used by the IBM Security Identity Manager server for subforms. Do not modify.</td>
</tr>
<tr>
<td>tenant</td>
<td>This file used for the creation of a new tenant. Do not modify.</td>
</tr>
<tr>
<td>tmsMessages</td>
<td>This file contains all error messages. Do not modify.</td>
</tr>
<tr>
<td>TungstenTheme</td>
<td>This file sets display values for applet accessibility. Do not modify.</td>
</tr>
</tbody>
</table>

adhocreporting.properties

The adhocreporting.properties file supports the custom reporting module.

Table 15 defines the properties used to configure reporting.

Table 15. adhocreporting.properties properties

<table>
<thead>
<tr>
<th>Report Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>reportPageSize</td>
</tr>
</tbody>
</table>
### Table 15. adhocreporting.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| `reportPageSize`          | Indicates the number of rows that are displayed on each page of a PDF report. The maximum number of rows on a page must not exceed 45.  
Example (default):          | `reportPageSize=45`                                                                                                                                                                                                                                                                                     |
| `reportColWidth`          | Indicates the width, in centimeters (cm), of the report column in a PDF report output. You can adjust the size of all columns by modifying this value.  
Note: 2.54 cm equals 1 inch.  
Example (default):          | `reportColWidth=20`                                                                                                                                                                                                                                                                                     |
| `availableForNonAdministrators` | Specifies whether to synchronize access control item-related information during data synchronization.  
Set this value to true to enable non-administrators to run reports.  
Set this value to false to disable all functions related to non-administrator execution of reports, such as access control item data synchronization and setting report access control items on reports.  
Example:          | `availableForNonAdministrators=true`                                                                                                                                                                                                                                                                  |
| `enableDeltaSchemaEnforcer` | Specifies whether to synchronize any schema changes in reporting.  
Schema changes include new mappings that were created or existing mappings that were removed with the Schema Designer.  
When set to true, the Incremental Data Synchronizer manages the attributes that are mapped (changed) in the Schema Designer since the last full data synchronization was run.  
When set to false, the Incremental Data Synchronizer does not synchronize the attributes which are mapped (changed) since the last full data synchronization was run.  
Example (default):          | `enableDeltaSchemaEnforcer=false`                                                                                                                                                                                                                                                                 |
| `changelogEnabled`        |                                                                                                                                                                                                                                                                                                                                                       |
Table 15. adhocreporting.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>changelogEnabled</td>
<td>Specifies whether the Incremental Data Synchronizer is used. Values include:</td>
<td>changelogEnabled=false</td>
</tr>
<tr>
<td></td>
<td>• true – Incremental Data Synchronizer is configured</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• false – Incremental Data Synchronizer is not configured</td>
<td></td>
</tr>
<tr>
<td>changelogBaseDN</td>
<td>Specifies the DN in the directory where the change log is configured.</td>
<td>changelogBaseDN=cn=changelog</td>
</tr>
<tr>
<td>changeLogFetchSize</td>
<td>Specifies the number of change logs to be fetched at one time from the directory server.</td>
<td>changeLogFetchSize=200</td>
</tr>
<tr>
<td>maximumChangeLogsToSynchronize</td>
<td>Specifies the maximum number of change logs to be synchronized in a single use of the Incremental Data Synchronizer.</td>
<td>maximumChangeLogsToSynchronize=10000</td>
</tr>
<tr>
<td>changeLogsToAnalyzeBeforeSynchronization</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 15. adhocreporting.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>changeLogFetchSize</strong></td>
<td>Specifies the number of fetched change log entries to be analyzed before all analyzed entries are synchronized to the database.</td>
<td>changeLogFetchSize=500</td>
</tr>
<tr>
<td><strong>changeLogsToAnalyzeBeforeSynchronization</strong></td>
<td>Specifies the number of fetched change log entries to be analyzed before all analyzed entries are synchronized to the database.</td>
<td>changeLogsToAnalyzeBeforeSynchronization=20000</td>
</tr>
<tr>
<td><strong>maximumChangeLogsToSynchronize</strong></td>
<td>Specifies the maximum number of change log entries to be synchronized.</td>
<td>maximumChangeLogsToSynchronize=100000</td>
</tr>
</tbody>
</table>

For example, consider the following values:
- changeLogFetchSize=500
- changeLogsToAnalyzeBeforeSynchronization=20000
- maximumChangeLogsToSynchronize=100000

500 change log entries are considered one batch. After 20,000 change log entries (40 batches) are analyzed, data synchronization occurs. This process repeats until 100,000 entries are analyzed (5 synchronizations).

Setting this value to 0 or a negative value results in synchronization of all fetched change log entries.

Example (default):
- changeLogsToAnalyzeBeforeSynchronization=5000

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>enableChangelogPruning</strong></td>
<td>Specifies whether changelog entries need to be pruned after they are successfully synchronized. This property takes effect only for the SunOne Version 5.2 directory server. For the IBM Tivoli Directory Server, see its documentation about pruning changelog entries.</td>
<td>enableChangelogPruning=false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>itimAdminID</strong></td>
<td>Specifies the administrator ID required to run the Incremental Data Synchronizer in a z/OS® environment.</td>
<td>itimAdminID=myadminid</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>itimAdminCredential</strong></td>
<td>Specifies the IBM Security Identity Manager password required to run the Incremental Data Synchronizer in a z/OS® environment.</td>
<td>itimAdminCredential=myadmincredential</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>createIndex</strong></td>
<td>Specifies whether to create database indexes for frequently used database columns. If this property is set to true, reports are generated more quickly.</td>
<td>createIndex=true</td>
</tr>
<tr>
<td><strong>true</strong></td>
<td>Creates indexes for columns that are used by reporting. Enabling this value might increase the data synchronization time.</td>
<td>createIndex=true</td>
</tr>
<tr>
<td><strong>false</strong></td>
<td>Does not create indexes during data synchronization. Disabling this value might increase the time that is needed to generate reports.</td>
<td>createIndex=false</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>reportIndexes</td>
<td>Specifies a set of <code>&lt;ENTITY:(ATTR1 ORDER1, ATTR2 ORDER2, ...)&gt;</code> values on which indexes are created. Both single and compound indexes can be created with this property. If you are creating a single index, use the name of entity that you see in the report designer or schema mapping. If you are defining a compound index, specify the exact table name, such as <code>Account</code> or <code>Person.cn</code>, instead of the entity name. You can specify an optional order <code>asc</code> or <code>desc</code> for an index. Observe the usage of a semi-colon as the delimiter between indexes. You must maintain the syntax of this property correctly, or indexes might not get created successfully. If you add additional indexes, follow the syntax for these default indexes: <code>reportIndexes=Person.cn asc;Account:eraccountcompliance;Account:(eraccountstatus asc);Account:erlastaccessdate asc;Account:eruid asc;Service:(servicetype asc);Service:erservicename asc;ProvisioningPolicy:erpolicyitemname asc;ProvisioningPolicy:erpolicytarget asc;ProvisioningPolicy:erpolicymembership asc;Role:errolename asc;Account:(eraccountstatus asc, erservice asc);Person_cn:(dn, cn);Account_owner:(dn asc, owner asc)</code></td>
<td></td>
</tr>
<tr>
<td>sqlBatchSize</td>
<td>Indicates the size of batch updates that are processed during data synchronization. To improve performance, set this value to a larger number. This value is affected by the specific database settings for the transaction log file size, a database property. Setting the value too high might cause data synchronization to fail. Always use the default value of 50 to avoid data synchronization failure. A value of 0, or a negative value, causes all SQL updates to be processed as a single batch. Example (default): <code>sqlBatchSize=50</code></td>
<td></td>
</tr>
<tr>
<td>attribsSkippedInSchema</td>
<td>These attributes contain XMLs as data. The reporting engine currently does not support reporting on these attributes. Example (on one line): <code>attribsSkippedInSchema=erEntitlements erAcl erHistoricalPassword erJavascript erLostPasswordAnswer erPassword erPlacementRule erxforms erXML</code></td>
<td></td>
</tr>
<tr>
<td>reportsAllowedAttributes</td>
<td>A set of attributes on which reporting engine does not enforce attribute-level access control. Example (default): <code>reportsAllowedAttributes=servicetype</code></td>
<td></td>
</tr>
<tr>
<td>reportsAllowedEntities</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 15. `adhocreporting.properties` properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>reportsAllowedEntities</code></td>
<td>A set of entities on which reporting engine does not enforce attribute-level access control.</td>
</tr>
<tr>
<td>Example (default):</td>
<td><code>reportsAllowedEntities=RecertificationPolicy, Group</code></td>
</tr>
<tr>
<td><code>reservedWords</code></td>
<td>Database reserved words. These words are not used as table/column names during Schema Mapping and Data Synchronization.</td>
</tr>
<tr>
<td>Example (on one line):</td>
<td><code>reservedWords=ALL ADD ALTER BACKUP BEGIN BY BULK CASCADE CHECK CHECKPOINT CLUSTORED COLUMN CREATE CURRENT DUMMY DOMAIN DELETE DEFAULT DISTINCT DROP FOREIGN FROM GROUP IDENTITY IDENTITY_INSERT IDENTITYCOL INSERT IN LIKE SET SELECT TABLE VALUES ORDER UID WHERE</code></td>
</tr>
<tr>
<td><code>disallowedChars</code></td>
<td>Characters that are not part of Table/Column name in database. If the entity/attribute name contains one or more of these characters, the characters are removed from the table or column name. The double backslashes (<code>\</code>) are used as escape characters.</td>
</tr>
<tr>
<td>Example (default):</td>
<td>`disallowedChars=~!@#%^&amp;*()+{}</td>
</tr>
<tr>
<td><code>disallowedCharsForStart</code></td>
<td>Characters are not used as the starting character of table or column name. The double backslashes (<code>\</code>) are used as escape characters.</td>
</tr>
<tr>
<td>Example (default):</td>
<td>`disallowedCharsForStart=~!@#$%^&amp;*()_+{}</td>
</tr>
<tr>
<td><code>maxTableNameLength</code></td>
<td>Default maximum length for a table name.</td>
</tr>
<tr>
<td>Example (default):</td>
<td><code>maxTableNameLength=30</code></td>
</tr>
<tr>
<td><code>maxColumnNameLength</code></td>
<td>Default maximum length for a column name.</td>
</tr>
<tr>
<td>Example (default):</td>
<td><code>maxColumnNameLength=30</code></td>
</tr>
<tr>
<td><code>maxTableNameLength_DB2</code></td>
<td>Maximum name length for a table name in DB2.</td>
</tr>
<tr>
<td>Example (default):</td>
<td><code>maxTableNameLength_DB2=128</code></td>
</tr>
<tr>
<td><code>maxColumnNameLength_DB2</code></td>
<td>Maximum name length for a column name in DB2.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>maxColumnNameLength_DB2</code></td>
<td>Maximum name length for a column name in DB2.</td>
</tr>
<tr>
<td><code>maxTableNameLength_ZDB2</code></td>
<td>Maximum name length for a table name in DB2 z/OS.</td>
</tr>
<tr>
<td><code>maxColumnNameLength_ZDB2</code></td>
<td>Maximum name length for a column name in DB2 z/OS.</td>
</tr>
<tr>
<td><code>maxTableNameLength_ORACLE=30</code></td>
<td>Maximum name length for a table name in Oracle.</td>
</tr>
<tr>
<td><code>maxColumnNameLength_ORACLE</code></td>
<td>Maximum name length for a column name in Oracle.</td>
</tr>
<tr>
<td><code>maxTableNameLength_MS_SQL_SERVER</code></td>
<td>Maximum name length for a table name in Microsoft SQL Server.</td>
</tr>
<tr>
<td><code>maxColumnNameLength_MS_SQL_SERVER</code></td>
<td>Maximum name length for a column name in Microsoft SQL Server.</td>
</tr>
</tbody>
</table>

 populateGroupMembers
Table 15. adhocreporting.properties properties (continued)

<table>
<thead>
<tr>
<th>Specifies whether Service group membership changes need to be synchronized during incremental synchronization. Service group membership information is stored in the GROUPMEMBERS table.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid values for this property are:</td>
</tr>
<tr>
<td>• true – Synchronizes membership changes to service groups and accesses (for example, because of a new access request).</td>
</tr>
<tr>
<td>• false – Does not synchronize group membership changes, since this type of synchronization is performance intensive.</td>
</tr>
<tr>
<td>Example (default):</td>
</tr>
<tr>
<td>populateGroupMembers=false</td>
</tr>
</tbody>
</table>

**CustomLabels.properties**

The property key and value pairs in the CustomLabels.properties file are used by the IBM Security Identity Manager user interface to display the label text for forms.

The key name must be entirely lowercase in each property key and value pair.

A separate CustomLabels.properties file exists for each individual language supported by IBM Security Identity Manager.

This file is used to provide localized versions of graphical user interface elements when IBM Security Identity Manager is installed in international environments.

Add the property key and value pairs in the CustomLabels.properties properties file to display any labels. To add labels to an access type hierarchy, you can use a period (.) as a separator to separate one or more access types. Use the period (.) separator only when there is more than one access type. For example, define as Application.Finance = Application Finance in the CustomLabels.properties properties file.

A file name extension identifies the specific language. For example:

**English**

    CustomLabels_en.properties

**Japanese**

    CustomLabels_ja.properties

**DataBaseFunctions.conf**

The custom reporting feature of IBM Security Identity Manager allows you to use database functions when designing custom report templates.

This file is in the ISIM_HOME/data directory.

You can use the database functions with the Report Designer component of IBM Security Identity Manager by defining the functions in the DataBaseFunctions.conf file.
Pre-defined database function properties use the following format in the DataBaseFunctions.conf file:

\(<function\_name>-<number\_of\_arguments>\)

Database users can also create and define functions for their custom use. Custom functions are called user-defined functions in Microsoft SQL and IBM DB2. Functions created as stored procedures in DB2 can also be used for reporting. Functions must be created with the database utilities that are provided by the respective database vendor.

User-defined database function properties use the following format in the DataBaseFunctions.conf file:

\(\text{user:}<function\_name>-<number\_of\_arguments>\)

Only functions with a single argument are supported in the IBM Security Identity Manager Report Designer.

Table 16. DataBaseFunctions.conf

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper</td>
<td>Converts the argument to uppercase. Example: Upper - 1</td>
</tr>
<tr>
<td>Lower</td>
<td>Converts the argument to lowercase. Example: Lower - 1</td>
</tr>
</tbody>
</table>

---

**enroleAuditing.properties**

The property key and value pairs in the enroleAuditing.properties file are used to enable or disable the tracking of changes made by an IBM Security Identity Manager user to business objects such as person, location, service, and other objects, or configuration of the system.

Any user request to change the IBM Security Identity Manager directory store or database can be audited and published in a report.

The following is a comprehensive list of events audited:

- ACI Management (Add, Add Authorization Owner, Delete, Delete Authorization Owner, Modify)
- Account Management (Add, Adopt, Change Password, Delete, Modify, Orphan, Password Pickup, Restore, Suspend, Synchronize Password)
- Access Management (Add, Remove)
- Access Configuration (Add, Remove, Modify)
- Authentication (Authenticate ITIM user)
- Container Management (Add, Delete, Modify)
- Delegate Authority (Add, Delete, Modify)
- Entitlement Workflow Management (Add, Delete, Modify)
• Entity Operation Management (Add, Delete, Modify)
• IBM Security Identity Manager Configuration (Add, Delete, Enforce, Install Profile, Modify, Uninstall Profile)
• Group Management (Add, Add Member, Delete, Modify, Remove Member)
• Migration (Agent Profile Install, Start Export, Start Import, Stop Export, Stop Import)
• Role Management (Add, Add Member, Delete, Modify, Remove Member)
• Person Management (Add, Delete, Modify, Restore, Self Register, Suspend, Transfer)
• Policy Management (Add, Commit Draft, Delete, Enforce Entire Policy, Modify, Save as Draft, Add Account Template, Change Account Template, Remove Account Template)
• Reconciliation (Run Recon, Set Recon Unit, Set Service Recon Parameters)
• Runtime Events (Start IBM Security Identity Manager, Stop IBM Security Identity Manager)
• Self Password Change (Change Password, Reset Password)
• Service Management (Add, Add Adoption Rule, Delete, Delete Adoption Rule, Modify, ModifyAdoption Rule)
• Service Policy Enforcement (Correct Non-Compliant, Mark Non-Compliant, Suspend Non-Compliant, Use Global Setting, Use Workflow For Non-Compliant)

Audited information specifically includes:
• The identity of the user who takes the action.
• The time the action was taken.
• The type of action taken.
• The data effected by the action.

Table 17 defines the properties used to configure how the auditing feature behaves.

<table>
<thead>
<tr>
<th>Table 17. enroleAuditing.properties properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IBM Security Identity Manager audit configuration settings</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Table 17. enroleAuditing.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.auditing.retrycount</td>
<td>The number of times auditing is tried again in case of failure. Valid values include any integer.</td>
<td>itim.auditing.retrycount=1</td>
</tr>
<tr>
<td>enrole.auditing.retrydelay</td>
<td>The wait time in milliseconds before trying again.</td>
<td>itim.auditing.retrydelay=5000</td>
</tr>
<tr>
<td>enrole.auditing.errorpopup.enabled</td>
<td>Enables or disables the process failure display.</td>
<td>enrole.auditing.errorpopup.enabled=false</td>
</tr>
<tr>
<td>enrole.auditing.errorpopup.fields</td>
<td>The process failure display always contains these attributes and their values: (name, subject, type, result_summary) You can additionally specify one or more of these attributes: (subject, comments, name, type, requester_type, requester_name, description, scheduled, started, completed, lastmodified, submitted, state, notify, requestee_name, subject_profile, subject_service, result_summary, result_detail) Example: enrole.auditing.errorpopup.fields=subject, comments</td>
<td>enrole.auditing.errorpopup.fields=subject, comments</td>
</tr>
<tr>
<td>enrole.auditing.errorpopup.textwrap</td>
<td>Specifies whether the text wraps at the end of the display.</td>
<td>enrole.auditing.errorpopup.textwrap=false</td>
</tr>
<tr>
<td>enrole.auditing.pageSize</td>
<td>Specifies the page size in lines that displaying unsuccessful processes or activities on the failed activity popup.</td>
<td>enrole.auditing.pageSize=10</td>
</tr>
<tr>
<td>enrole.auditing.pageLinkMax</td>
<td>Specifies the number of page links for multi-page result sets on the failed activity.</td>
<td>enrole.auditing.pageLinkMax=10</td>
</tr>
<tr>
<td>enrole.auditing.viewRequests.skipServiceLookup</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 17. enroleAuditing.properties properties (continued)

<table>
<thead>
<tr>
<th>Description</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not change this property key and value unless you are a qualified administrator.</td>
<td>enrole.auditing.viewRequests.skipServiceLookup.customProcessTypes=CP</td>
</tr>
<tr>
<td>Specifies the custom process type that does not have a service or an account as subject data in the input parameters of its corresponding workflow operation. To use this property, add it to the $ISIM_HOME/data/enroleAuditing.properties file with a custom process type value.</td>
<td>Valid values: A comma-separated custom process type value.</td>
</tr>
<tr>
<td>Example (default):</td>
<td></td>
</tr>
<tr>
<td>enrole.auditing.viewRequests.skipServiceLookup.customProcessTypes=CP</td>
<td></td>
</tr>
</tbody>
</table>

**enRoleAuthentication.properties**

The enRoleAuthentication.properties file specifies the type of method that is used by the Security Identity Manager Server to authenticate users and identifies the Java object that provides the specified authentication mechanism.

Additionally, the file specifies objects that support Tivoli Access Manager WebSEAL single sign-on and administration of IBM Security Identity Manager to managed remote services.

Authentication properties are specified with a property key and value pair format:

```
property-key-name=value
```

The `property-key-name` is an identifier for the authentication mechanism or resource. The `value` is the name of the Java object that provides the authentication service, expressed also as a key and value pair.

```
factory=value
```

The `factory` key name represents a special category for authentication support within the IBM Security Identity Manager software. The `value` is the actual name of the Java object.

For example (entered on one line):

```
enrole.authentication.provider.service=
factory=com.ibm.enrole.authentication.service.
ServiceAuthenticationProviderFactory
```

Table 18 defines the properties used to configure IBM Security Identity Manager authentication.

Table 18. enRoleAuthentication.properties properties

<table>
<thead>
<tr>
<th>Authentication method</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.authentication.requiredCredentials={simple}</td>
</tr>
<tr>
<td>Table 18. enRoleAuthentication.properties properties (continued)</td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>Specifies the required authentication method for users who log in to the IBM Security Identity Manager Server.</td>
</tr>
<tr>
<td>The valid value for this property is:</td>
</tr>
<tr>
<td>• simple - User name and password.</td>
</tr>
<tr>
<td>Example (default):</td>
</tr>
<tr>
<td>enrole.authentication.requiredCredentials=simple</td>
</tr>
</tbody>
</table>

### Authentication providers (factories)

**enrole.authentication.provider.simple**

Specifies the Java object that handles authentication with user name and password.

Example (entered on a single line):

```
enrole.authentication.provider.simple=\  
  factory=com.ibm.itim.authentication.simple.  
  SimpleAuthenticationProviderFactory
```

### Authentication service provider

**enrole.authentication.provider.service**

Specifies the Java object that transparently handles IBM Security Identity Manager access to managed remote services and to manage changes in the accounts to these remote services.

These changes include addition, deletion, suspension, restoration, and modification of accounts on the remote service. When you log in to IBM Security Identity Manager, you can change the login and password information for an account on the managed remote service.

The `ServiceAuthenticationProviderFactory` mechanism works with the agent for a given remote service and processes the changed information.

Example (entered on a single line):

```
enrole.authentication.provider.service=\  
  factory=com.ibm.itim.authentication.service.  
  ServiceAuthenticationProviderFactory
```

### WebSEAL single sign-on

**enrole.authentication.provider.webseal**

Specifies the Java object that allows single sign-on in a WebSEAL environment.

Example (entered on a single line):

```
enrole.authentication.provider.webseal=\  
  factory=com.ibm.itim.authentication.webseal.WebsealProviderFactory
```

**enrole.authentication.idsEqual**
Table 18. enRoleAuthentication.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.authentication.idsEqual</td>
<td>Indicates the appropriate algorithm for mapping the Tivoli Access Manager user ID to an IBM Security Identity Manager user ID. An internal identity mapping algorithm is used to ensure the success of the single sign-on operation. Valid values for this property are:</td>
</tr>
<tr>
<td>true</td>
<td>The Tivoli Access Manager user ID is the same as the IBM Security Identity Manager user ID.</td>
</tr>
<tr>
<td>false</td>
<td>The Tivoli Access Manager user ID is not the same as the IBM Security Identity Manager user ID.</td>
</tr>
<tr>
<td>Example: enrole.authentication.idsEqual=true</td>
<td></td>
</tr>
</tbody>
</table>

enRoleDatabase.properties

The enRoleDatabase.properties file specifies attributes that support the relational database used by IBM Security Identity Manager.

The property key values contained in this file are synchronized with values in the appropriate application server configuration file. Most values in this file are supplied during initial installation of IBM Security Identity Manager and the configuration of the database. You can make subsequent changes to some values. However, you must use the runConfig utility to synchronize the property file values with the values in the application server configuration file.

IBM Security Identity Manager uses Java Database Connectivity (JDBC) to access the relational database. With the JDBC API, you can access virtually any tabular data source from the Java programming language.

Table 19 defines the properties used to configure database properties.

Table 19. enRoleDatabase.properties properties

<table>
<thead>
<tr>
<th>Database information</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>database.db.type</td>
<td>Do not modify this property key. The value is supplied during the initial installation of IBM Security Identity Manager. Specifies the database type that is used by IBM Security Identity Manager. Example: database.db.type=DB2</td>
</tr>
<tr>
<td>database.db.server</td>
<td></td>
</tr>
</tbody>
</table>
Table 19. enRoleDatabase.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| database.db.server     | Specifies the name or local alias name of the remote database. The value is supplied during the installation of IBM Security Identity Manager and the configuration of the database. The value must be provided in the format: `db_host_name:port:database_name`. Examples:  
  - DB2: `10.77.214.35:50000:itimdb` 
  - Oracle: `tivsun13:1521:itimdb` 
  - Microsoft SQL Server: `tivsun13:1433:itimdb` |
| database.db.owner      | Do not modify this property key. The value is built in to the system. Specifies the name of the database schema owner for IBM Security Identity Manager. Example (default): 
  `database.db.owner=itimuser` |
| database.db.user       | Do not modify this property key. The value is built in to the system. Specifies a default database user for IBM Security Identity Manager. Example (default): 
  `database.db.user=itimuser` |
| database.db.password   | Do not modify this property key. The value is supplied during database configuration. Specifies the password for the database user. Encryption of this value is specified by the `enrole.password.database.encrypted` property in `enRole.properties`. The password value is encrypted by default unless the encryption setting was deactivated with the `runConfig` utility. Example: 
  `database.db.password=secret` |
### Table 19. enRoleDatabase.properties properties (continued)

<table>
<thead>
<tr>
<th><strong>Connection pool properties</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>database.jdbc.connectionPool.initialCapacity</strong></td>
</tr>
<tr>
<td>Do not manually edit this file to modify this property key value. Use the runConfig utility to change this value.</td>
</tr>
<tr>
<td>Specifies the initial number of physical database connections to create for the connection pool. This value must be less than or equal to the <code>database.jdbc.connectionPool.maxCapacity</code> value.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
</tr>
<tr>
<td><code>database.jdbc.connectionPool.initialCapacity=5</code></td>
</tr>
<tr>
<td><strong>database.jdbc.connectionPool.maxCapacity</strong></td>
</tr>
<tr>
<td>Do not manually edit this file to modify this property key value. Use the runConfig utility to change this value.</td>
</tr>
<tr>
<td>Specifies the maximum number of physical database connections that can be created. This value is used to manage system performance tuning.</td>
</tr>
<tr>
<td><strong>Example (default)</strong></td>
</tr>
<tr>
<td><code>database.jdbc.connectionPool.maxCapacity=50</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>JDBC driver</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>database.jdbc.driverurl</strong></td>
</tr>
<tr>
<td>Do not remove or modify this property key and value.</td>
</tr>
<tr>
<td>Specifies the URL of the JDBC driver. The default value is <code>jdbc:db2://db_host_name:port/database_name</code>.</td>
</tr>
<tr>
<td><strong>Examples</strong></td>
</tr>
<tr>
<td>- <strong>DB2</strong></td>
</tr>
<tr>
<td><code>jdbc:db2://10.77.214.31:50000/itimdb</code></td>
</tr>
<tr>
<td>- <strong>Oracle</strong></td>
</tr>
<tr>
<td><code>jdbc:oracle:thin:@host_name:1521:itimdb</code></td>
</tr>
<tr>
<td>- <strong>Microsoft SQL Server</strong></td>
</tr>
<tr>
<td><code>jdbc:sqlserver://;server=9.72.121.180;port=1433;database=itimdb</code></td>
</tr>
</tbody>
</table>

| **database.jdbc.driver** |
Table 19. enRoleDatabase.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>database.jdbc.driver</td>
<td>Specifies the JDBC driver name.</td>
<td>com.ibm.db2.jcc.DB2Driver, oracle.jdbc.driver.OracleDriver, com.microsoft.sqlserver.jdbc.SQLServerDriver</td>
</tr>
</tbody>
</table>

---

**enRoleLDAPConnection.properties**

The enRoleLDAPConnections.properties file provides standard configuration settings that allow successful communication between IBM Security Identity Manager and the LDAP directory server.

Table 20 defines the properties used to configure LDAP directory server properties.

### Table 20. enRoleLDAPConnection.properties properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.naming.factory.initial</td>
<td>Specifies the built-in Java class file that provides the communication interface between IBM Security Identity Manager and the LDAP directory server. The Java Naming and Directory Interface (JNDI) protocol is used.</td>
<td>com.sun.jndi.ldap.LdapCtxFactory</td>
</tr>
</tbody>
</table>

**LDAP context:** Context.INITIAL_CONTEXT_FACTORY

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.naming.provider.url</td>
<td>Specifies the URL of the LDAP directory server. The LDAP server is on:</td>
<td>ldap://localhost:389</td>
</tr>
</tbody>
</table>

**LDAP context:** Context.PROVIDER_URL

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.naming.security.principal</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 20. enRoleLDAPConnection.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
<th>Encryption Details</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.naming.security.principal</td>
<td>Specifies the distinguished name (DN) of the LDAP administration account on the LDAP directory server.</td>
<td>java.naming.security.principal=cn=root</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The value for this key is initially configured during IBM Security Identity Manager installation. You can also provide this value with the <code>ldapconfig</code> utility or <code>runConfig</code> utility.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example for Sun Open Net Environment (ONE) Directory Server:</td>
<td>java.naming.security.principal=cn=directory manager</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LDAP context: Context.SECURITY_PRINCIPAL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>java.naming.security.credentials</td>
<td>Specifies the password for the LDAP administration account on the LDAP directory server.</td>
<td>java.naming.security.credentials=ibmldap</td>
<td>The encryption type is initially configured during IBM Security Identity Manager installation.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The value for this key is initially configured during IBM Security Identity Manager installation. You can also provide this value with the <code>ldapconfig</code> utility or <code>runConfig</code> utility.</td>
<td></td>
<td>The encryption type is configured by the <code>enrole.password.ldap.encrypted</code> property in the <code>enRole.properties</code> file.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>java.naming.security.credentials=ibmldap</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LDAP context: Context.SECURITY_CREDENTIALS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>java.naming.security.protocol</td>
<td>By default, this property is commented out.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies the protocol that is used for communication between IBM Security Identity Manager and the LDAP directory server. For example, to enable SSL, uncomment the line and change it to <code>java.naming.security.protocol=ssl</code>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LDAP context: Context.SECURITY_PROTOCOL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>java.naming.security.authentication</td>
<td>Do not modify this property key and value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies the authentication type that is used by the LDAP directory server. Valid types include:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <code>none</code> - The anonymous: user becomes a member of an unauthenticated group.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <code>simple</code> - The user supplies a user name and password.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <code>strong</code> - A stronger authentication mechanism that you provide.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>java.naming.security.authentication=simple</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LDAP context: Context.SECURITY_AUTHENTICATION</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 20. enRoleLDAPConnection.properties properties (continued)

### java.naming.referral

Do not modify this property key and value.

If multiple LDAP directory servers are linked in the IBM Security Identity Manager environment, this property specifies whether to use links when a referral is needed to complete a request for LDAP information.

Valid values include:
- **follow** — Use links to complete an LDAP information request.
- **ignore** — Do not use links to complete an LDAP information request.
- **throw** — Do not use links to complete an LDAP information request and return an error message.

Example:
```
java.naming.referral=follow
```

LDAP context: Context.REFERRAL

### java.naming.batchsize

Do not modify this property key and value.

A JNDI property that specifies the number of data elements returned at one time during a request (query) to the LDAP directory server. A larger number reduces the number of LDAP fetches, which might improve performance.

A value of 0 blocks any control by the client (IBM Security Identity Manager) until all requested elements are returned.

Example:
```
java.naming.batchsize=100
```

LDAP context: Context.BATCHSIZE

### java.naming.ldap.derefAliases

Specifies that look up for an object by using the alias dereferences the alias so that what is returned is the object pointed to by the DN of the alias.

Valid values include:
- **never** — Do not dereference an alias during object lookup.
- **always** — Dereference an alias during object lookup.
- **finding** — Dereference an alias during object lookup (only during name resolution).
- **searching** — Dereference an alias during object lookup (only after name resolution).

Example:
```
java.naming.ldap.derefAliases=never
```

### java.naming.ldap.attributes.binary

Chapter 15. Supplemental property files
### enRoleLDAPConnection.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not modify this property key and value.</td>
<td>Specifies IBM Security Identity Manager attributes that are treated as binary data type. Multiple attribute values are separated by a single space.</td>
</tr>
</tbody>
</table>
| Example (on a single line):                   |java.naming.ldap.attributes.binary=erPassword  
erHistoricalPassword erSynchPassword erServicePassword erPersonPassword            |
| LDAP context: attribute.binary                |                                                                                                  |
| com.sun.jndi.ldap.connect.pool                |Activates the LDAP connection pool.                                                               |
| Valid values include:                         |• true - Use the LDAP connection pool.                                                             |
| • false - Do not use the LDAP connection pool. |                                                                                                  |
| Example (default):                            |com.sun.jndi.ldap.connect.pool=true                                                               |
| com.sun.jndi.ldap.connect.timeout             |Specifies the number of milliseconds that a client waits for a pooled connection to become available. If the property is not specified, the client waits indefinitely. |
| Example:                                      |#com.sun.jndi.ldap.connect.timeout=                                                               |

### enRoleLogging.properties

The enRoleLogging.properties file specifies attributes that govern the operation of the jlog logging and tracing API that is bundled with IBM Security Identity Manager.

jlog is a logging package for Java. With this package, you can log messages by message type and priority. At run time, you also can control how these messages are formatted and where they are reported.

Table 21 defines the properties used to configure IBM Security Identity Manager logging properties.

### enRoleLogging.properties properties

<table>
<thead>
<tr>
<th>General settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logger.refreshInterval</td>
<td>Specifies the refresh interval [in milliseconds] of the logging properties.</td>
</tr>
<tr>
<td>Example:</td>
<td>logger.refreshInterval=300000</td>
</tr>
<tr>
<td>logger.msg.com.ibm.itim.security.logChoice</td>
<td></td>
</tr>
<tr>
<td>Table 21. enRoleLogging.properties properties (continued)</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Specifies the type of authentication attempts to log.</td>
<td></td>
</tr>
<tr>
<td>Valid values are:</td>
<td></td>
</tr>
<tr>
<td>• <strong>failure</strong> — Log authentication failures.</td>
<td></td>
</tr>
<tr>
<td>• <strong>success</strong> — Log authentication successes.</td>
<td></td>
</tr>
<tr>
<td>• <strong>both</strong> — Log both authentication failures and successes.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>logger.msg.com.ibm.itim.security.logChoice=failure</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>logger.msg.com.ibm.itim.security.logging</td>
<td></td>
</tr>
<tr>
<td>Specifies whether authentication attempts are logged or not.</td>
<td></td>
</tr>
<tr>
<td>Valid values are:</td>
<td></td>
</tr>
<tr>
<td>• <strong>true</strong> — Log authentication attempts.</td>
<td></td>
</tr>
<tr>
<td>• <strong>false</strong> — Do not log authentication attempts.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>logger.msg.com.ibm.itim.security.logging=true</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>handler.file.security.maxFiles</td>
<td></td>
</tr>
<tr>
<td>Specifies the maximum number of security log files.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>handler.file.security.maxFiles=10</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>logger.msg.level</td>
<td></td>
</tr>
<tr>
<td>Specifies the logging level for messages.</td>
<td></td>
</tr>
<tr>
<td>Valid values are:</td>
<td></td>
</tr>
<tr>
<td>• <strong>INFO</strong></td>
<td></td>
</tr>
<tr>
<td>• <strong>WARN</strong></td>
<td></td>
</tr>
<tr>
<td>• <strong>ERROR</strong></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>logger.msg.level=INFO</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>handler.file.msg.maxFiles</td>
<td></td>
</tr>
<tr>
<td>Specifies the maximum number of message log files.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>handler.file.msg.maxFiles=5</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>logger.trace.level</td>
<td></td>
</tr>
</tbody>
</table>
Table 21. enRoleLogging.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logger.trace.level</td>
<td>Specifies the tracing level. The supported trace levels are:</td>
</tr>
<tr>
<td></td>
<td>- DEBUG_MIN</td>
</tr>
<tr>
<td></td>
<td>- DEBUG_MID</td>
</tr>
<tr>
<td></td>
<td>- DEBUG_MAX</td>
</tr>
<tr>
<td></td>
<td>DEBUG_MAX is the most verbose trace level and can effect system performance.</td>
</tr>
<tr>
<td></td>
<td>When you debug a problem, avoid setting DEBUG_MAX at logger.trace. Set the</td>
</tr>
<tr>
<td></td>
<td>DEBUG_MAX at the effected components or packages.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>logger.trace.level=DEBUG_MIN</td>
</tr>
<tr>
<td>handler.file.trace.maxFiles</td>
<td>Specifies the maximum number of trace log files.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>handler.file.trace.maxFiles=10</td>
</tr>
<tr>
<td>handler.file.maxFileSize</td>
<td>Specifies the maximum log file size in kilobytes</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>handler.file.maxFileSize=1024</td>
</tr>
<tr>
<td>Logger root properties</td>
<td></td>
</tr>
<tr>
<td>jlog.noLogCmd</td>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td></td>
<td>Disables the log command server.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>jlog.noLogCmd=true</td>
</tr>
<tr>
<td>logger.className</td>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td></td>
<td>Specifies the class name of the logger.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>logger.className=com.ibm.log.PDLogger</td>
</tr>
<tr>
<td>logger.description</td>
<td>Specifies the description of the logger.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>logger.description=TIM PD Logger</td>
</tr>
<tr>
<td>logger.product</td>
<td></td>
</tr>
</tbody>
</table>
## Table 21. enRoleLogging.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>logger.product</td>
<td>Specifies the product name.</td>
<td>logger.product=CTGIM</td>
</tr>
<tr>
<td>logger.productInstance</td>
<td>Specifies the server instance name. The value is supplied during the</td>
<td>logger.productInstance=myserver</td>
</tr>
<tr>
<td></td>
<td>installation of IBM Security Identity Manager.</td>
<td></td>
</tr>
<tr>
<td>logger.msg.description</td>
<td>Specifies the description of the message logger.</td>
<td>logger.msg.description=TIM PD Message Logger</td>
</tr>
<tr>
<td>logger.msg.logging</td>
<td>Turns logging on or off for messages. Valid values are:</td>
<td>logger.msg.logging=true</td>
</tr>
<tr>
<td></td>
<td>• true — Turns logging on.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• false — Turns logging off.</td>
<td></td>
</tr>
<tr>
<td>logger.msg.messageFile</td>
<td>Specifies the resource bundle name of localizable messages.</td>
<td>logger.msg.messageFile=tmsMessages</td>
</tr>
<tr>
<td>logger.msg.listenerNames</td>
<td>Do not modify this property key and value.</td>
<td></td>
</tr>
</tbody>
</table>

---

**Chapter 15. Supplemental property files**
Table 21. enRoleLogging.properties properties (continued)

### Security logger properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>logger.msg.com.ibm.itim.security.listenerNames</td>
<td>Specifies the listener names attached to the security logger.</td>
<td>Do not modify this property key and value. logger.msg.com.ibm.itim.security.listenerNames=handler.file.security</td>
</tr>
</tbody>
</table>

### Trace logger properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>logger.trace.description</td>
<td>Specifies the description of the trace logger.</td>
<td>Do not modify this property key and value. logger.trace.description=TIM PD Trace Logger</td>
</tr>
<tr>
<td>logger.trace.logging</td>
<td>Turns trace logging on or off. Valid values are:</td>
<td>Do not modify this property key and value. logger.trace.logging=true</td>
</tr>
<tr>
<td></td>
<td>• true — Turns logging on.</td>
<td>logger.trace.logging=true</td>
</tr>
<tr>
<td></td>
<td>• false — Turns logging off.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>logger.trace.listenerNames</td>
<td>Specifies the listener names attached to the trace logger.</td>
<td>Do not modify this property key and value. logger.trace.listenerNames=handler.file.trace</td>
</tr>
<tr>
<td>logger.trace.com.ibm</td>
<td></td>
<td>logger.trace.com.ibm</td>
</tr>
</tbody>
</table>
Table 21. enRoleLogging.properties properties (continued)

Edit the level of these component loggers to adjust the amount of tracing information written to the trace log.

The supported trace levels are:
- DEBUG_MIN
- DEBUG_MID
- DEBUG_MAX

Component loggers are:

Note: The `logger.trace.com.ibm.itim.script.level` component logger is equivalent to `logger.trace.com.ibm.itim.fesiextensions.level` (deprecated).

```
<table>
<thead>
<tr>
<th>Logger Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```

Applet tracing properties

```
logger.trace.com.ibm.itim.applet.logging
```

Enables or disables applet trace logging.

Example:
```
logger.trace.com.ibm.itim.applet.logging=true
```

```
logger.trace.com.ibm.itim.applet.level
```

 Specifies the applet tracing level.

The supported trace levels are:
- DEBUG_MIN
- DEBUG_MID
- DEBUG_MAX

Example:
```
logger.trace.com.ibm.itim.applet.level=DEBUG_MIN
```
### File handler properties

**handler.file.className**
- Do not modify this property key and value.
- Specifies the class name of the file handler.
- Example:
  ```
  handler.file.className=com.ibm.log.FileHandler
  ```

**handler.file.description**
- Specifies the description of the file handler.
- Example:
  ```
  handler.file.description=TIM File Handler
  ```

**handler.file.fileDir**
- Do not modify this property key and value.
- Specifies the base directory of the file handler. This value is supplied during installation.
- Example:
  ```
  handler.file.fileDir=c:/tivoli_comm_dir/CTGIM/logs
  ```

**handler.file.formatterName**
- Do not modify this property key and value.
- Specifies the formatter of the file handler.
- Example:
  ```
  handler.file.formatterName=formatter.PDXML
  ```

### Message logging file handler properties

**handler.file.msg.fileName**
- Specifies the message log file.
- Example:
  ```
  handler.file.msg.fileName=msg.log
  ```

**handler.file.msg.formatterName**
- Do not modify this property key and value.
- Specifies the formatter of the message file handler.
- Example:
  ```
  handler.file.msg.formatterName=formatter.PDXML
  ```

### Security logging file handler properties
Table 21. enRoleLogging.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>handler.file.security.fileDir</td>
<td>Specifies the security log directory.</td>
<td>handler.file.security.fileDir=c:/tivoli_comm_dir/CTGIM/logs</td>
</tr>
<tr>
<td>handler.file.security.fileName</td>
<td>Specifies the security log file.</td>
<td>handler.file.security.fileName=access.log</td>
</tr>
<tr>
<td>handler.file.security.formatterName</td>
<td>Do not modify this property key and value.</td>
<td>handler.file.security.formatterName=formatter.PDXML.security</td>
</tr>
<tr>
<td>Trace file handler properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>handler.file.trace.fileName</td>
<td>Specifies the trace file name.</td>
<td>handler.file.trace.fileName=trace.log</td>
</tr>
<tr>
<td>handler.file.trace.formatterName</td>
<td>Do not modify this property key and value.</td>
<td>handler.file.trace.formatterName=formatter.PDXML.trace</td>
</tr>
<tr>
<td>FFDC (First-Failure Data Capture) file copy handler properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>handler.ffdc.baseDir</td>
<td>Do not modify this property key and value.</td>
<td>handler.ffdc.baseDir=c:/tivoli_comm_dir/CTGIM/ffdc</td>
</tr>
<tr>
<td>handler.ffdc.triggerRepeatTime</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 21. enRoleLogging.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies the minimum time [in milliseconds] after an initial triggering that the handler responds to subsequent triggering events.</td>
<td>Example: handler.ffdc.triggerRepeatTime=300000</td>
<td>Do not modify this property key and value.</td>
<td></td>
</tr>
<tr>
<td>handler.ffdc.fileCopy.className</td>
<td>Do not modify this property key and value.</td>
<td>Specifies the handler class name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: handler.ffdc.fileCopy.className=com.tivoli.log.FileCopyHandler</td>
<td></td>
<td></td>
</tr>
<tr>
<td>handler.ffdc.fileCopy.triggerFilter</td>
<td>Specifies the filter to control which events trigger an FFDC action.</td>
<td>Example: handler.ffdc.fileCopy.triggerFilter=filter.msgId</td>
<td></td>
</tr>
<tr>
<td>handler.ffdc.fileCopy.fileTimestampFormat</td>
<td>Do not modify this property key and value.</td>
<td>Specifies the time stamp format which is appended to the FFDC folder name and file names.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: handler.ffdc.fileCopy.fileTimestampFormat=yyyy.MM.dd-HH.mm.ss</td>
<td></td>
<td></td>
</tr>
<tr>
<td>handler.ffdc.fileCopy.filesToCopy</td>
<td>Specifies the files to be copied to the FFDC directory when the FFDC is triggered.</td>
<td>Example (on a single line): handler.ffdc.fileCopy.filesToCopy= &quot;c:/tivoli_comm_dir/CTGIM/logs/trace.log&quot;  &quot;c:/tivoli_comm_dir/CTGIM/logs/msg.log&quot;</td>
<td></td>
</tr>
</tbody>
</table>

### FFDC message id filter properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter.msgId.className</td>
<td>Do not modify this property key and value.</td>
<td>Specifies the class name of the message ID filter.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: filter.msgId.className=com.tivoli.log.MsgIdFilter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>filter.msgId.description</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 21. enRoleLogging.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter.msgId.description</td>
<td>Specifies the description of the message ID filter.</td>
<td>filter.msgId.description=IBM Tivoli Identity Manager FFDC Message Id Filter</td>
</tr>
<tr>
<td>filter.msgId.msgIds</td>
<td>Specifies the TMS message IDs that trigger the FFDC action. The listed message IDs represent the most severe system errors.</td>
<td>filter.msgId.msgIds=CTGIMA401E CTGIMA438W CTGIME013E CTGIME035E CTGIME203E CTGIMF003E CTGIMF011E CTGIMF012E CTGIMF013E CTGIMF014E</td>
</tr>
<tr>
<td>filter.msgId.mode</td>
<td>Do not modify this property key and value.</td>
<td>filter.msgId.mode=PASSTHRU</td>
</tr>
<tr>
<td>filter.msgId.msgIdRepeatTime</td>
<td>Specifies the minimum time in milliseconds to wait after a log event is passed with a TMS message ID before it passes another one with the same ID.</td>
<td>filter.msgId.msgIdRepeatTime=300000</td>
</tr>
<tr>
<td>formatter.className</td>
<td>Specifies the class name of the formatter.</td>
<td>formatter.className=com.ibm.log.Formatter</td>
</tr>
<tr>
<td>formatter.description</td>
<td>Specifies the description of the formatter.</td>
<td>formatter.description=formatter</td>
</tr>
<tr>
<td>formatter.dateFormat</td>
<td>Specifies the Java SimpleDateFormat pattern to format event dates.</td>
<td>formatter.dateFormat=yyyy.MM.dd</td>
</tr>
<tr>
<td>formatter.timeFormat</td>
<td>Specifies the Java SimpleDateFormat pattern to format event dates.</td>
<td>formatter.timeFormat=yyyy.MM.dd</td>
</tr>
</tbody>
</table>
Table 21. enRoleLogging.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formatter.timeFormat</td>
<td>Specifies the Java SimpleDateFormat pattern to format event time.</td>
</tr>
<tr>
<td>Example:</td>
<td>formatter.timeFormat=HH:mm:ss.SSS</td>
</tr>
</tbody>
</table>

**PDXML formatter properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formatter.PDXML.className</td>
<td>Specifies the formatter class name which formatting log event in LOG XML format.</td>
</tr>
<tr>
<td>Example:</td>
<td>formatter.PDXML.className=com.ibm.itim.logging.LogXMLFormatter</td>
</tr>
<tr>
<td>formatter.PDXML.description</td>
<td>Specifies the description of the formatter.</td>
</tr>
<tr>
<td>Example:</td>
<td>formatter.PDXML.description=TIM Log XML Formatter</td>
</tr>
<tr>
<td>formatter.PDXML.msg.forceAsMessage</td>
<td>Force the message formatter to format all output as message events, regardless of their contents.</td>
</tr>
<tr>
<td>Example:</td>
<td>formatter.PDXML.msg.forceAsMessage=true</td>
</tr>
</tbody>
</table>

**Enabling tracing for the IBM Security Identity Manager user interface**

You must set the level to FINEST in WebSphere Application Server to get the user interface trace working.

Enabling tracing for the IBM Security Identity Manager user interface is a two-step process:
- Set the appropriate component loggers in the enRoleLogging.properties file.
- Enable WebSphere tracing by logging in to the WebSphere Application Server administrative console.

To enable the tracing level for the WebSphere Application Server administrative console, complete these steps:
1. Log in to the WebSphere Application Server administrative console.
2. Select **Troubleshooting -> Logs and Trace**.
3. Click the appropriate server (for example, **Server1**).
4. Select **Change log detail levels** under **General Properties**.
   - To make a static change to the configuration, click the **Configuration** tab.
     - A list of well-known components, packages, and groups is displayed.
   - To change the configuration dynamically, click the **Runtime** tab.
The list of components, packages, and groups displays all the components that are currently registered on the running server.

5. Expand the node for com.ibm.itim.* under *[All Components].
6. Click the node labeled com.ibm.itim.ui.* and select All Messages and Traces.
7. Click Apply.
8. Click OK.
9. Stop and then restart the WebSphere Application Server to set the static configuration changes.

You must enable the debug level for the user interface package in the following section of the enRoleLoggings.properties file:

```
# UI-tier tracing
# logger.trace.com.ibm.itim.ui.level=DEBUG_MIN
# logger.trace.com.ibm.itim.ui.common.level=DEBUG_MIN
# logger.trace.com.ibm.itim.ui.controller.level=DEBUG_MIN
# logger.trace.com.ibm.itim.ui.customizer.level=DEBUG_MIN
# logger.trace.com.ibm.itim.ui.help.level=DEBUG_MIN
# logger.trace.com.ibm.itim.ui.impl.level=DEBUG_MIN
# logger.trace.com.ibm.itim.ui.listener.level=DEBUG_MIN
# logger.trace.com.ibm.itim.ui.tasklauncher.level=DEBUG_MIN
# logger.trace.com.ibm.itim.ui.validator.level=DEBUG_MIN
# logger.trace.com.ibm.itim.ui.view.level=DEBUG_MIN
# logger.trace.com.ibm.itim.ui.viewmodel.level=DEBUG_MIN
```


---

**enRoleMail.properties**

The enRoleMail.properties file contains attributes that specify the mail transport protocol that is used by the JavaMail API and other IBM Security Identity Manager application-specific properties. You must provide the values for the application-specific properties.

Default values are provided for the JavaMail-specific properties (including the default mail provider and protocol). If you change the default values for the specific JavaMail properties, you must provide your own testing and verification of your custom protocol and implementation.

Go to the following URL for additional usage and provider information:

http://java.sun.com/products/javamail/

Table 22 defines the properties used to configure IBM Security Identity Manager mail properties.

<table>
<thead>
<tr>
<th>Mail attributes specific to the IBM Security Identity Manager application</th>
</tr>
</thead>
<tbody>
<tr>
<td>mail.baseurl</td>
</tr>
<tr>
<td>Property</td>
</tr>
<tr>
<td>--------------------------</td>
</tr>
<tr>
<td>mail.baseurl</td>
</tr>
<tr>
<td>mail.itim.context</td>
</tr>
<tr>
<td>mail.context.console</td>
</tr>
<tr>
<td>mail.context.enduser</td>
</tr>
<tr>
<td>mail.from</td>
</tr>
<tr>
<td>mail.transport.protocol</td>
</tr>
</tbody>
</table>
Table 22. enRoleMail.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mail.title</td>
<td>This property is not used to specify the title banner of email notification in IBM Tivoli Identity Manager Version 5.0.</td>
</tr>
<tr>
<td></td>
<td>The <strong>mail.title</strong> property of <strong>Labels.properties</strong> specifies title banner of email notification. You must edit this properties file directly to provide the value to this property.</td>
</tr>
<tr>
<td></td>
<td>This property was previously required.</td>
</tr>
<tr>
<td>Example (default) at previous releases:</td>
<td><strong>mail.title</strong>=ITIM notification</td>
</tr>
</tbody>
</table>

Mail attributes specific to the built-in JavaMail service

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mail.host</td>
<td>Specifies the IP address of the computer where the mail server is located.</td>
</tr>
<tr>
<td></td>
<td>This value is initially provided during IBM Security Identity Manager installation. You can also provide this value with the <strong>runConfig</strong> utility.</td>
</tr>
<tr>
<td></td>
<td>This property is required.</td>
</tr>
<tr>
<td>Example:</td>
<td><strong>mail.host</strong>=111.222.333.444</td>
</tr>
<tr>
<td>mail.protocol.host</td>
<td>Specifies the IP address of the protocol-specific default mail server. This property key overrides the mail.host property key.</td>
</tr>
<tr>
<td></td>
<td>By default, this property is not required and no value is provided.</td>
</tr>
<tr>
<td>mail.protocol.user</td>
<td>Specifies the protocol-specific default user name for connecting to the Mail server. This property key overrides the mail.user property key.</td>
</tr>
<tr>
<td></td>
<td>By default, this property is not required and no value is provided.</td>
</tr>
<tr>
<td>mail.protocol.class</td>
<td>Specifies the Java class implementation of the mail protocol.</td>
</tr>
<tr>
<td></td>
<td>Example (default): <strong>mail.SMTP.class</strong>=com.sun.mail.smtp.SMTPTransport</td>
</tr>
<tr>
<td>mail.store.protocol</td>
<td>Specifies the default message access protocol.</td>
</tr>
<tr>
<td></td>
<td>By default, this property is not required and no value is provided.</td>
</tr>
<tr>
<td>mail.user</td>
<td></td>
</tr>
</tbody>
</table>
Table 22. enRoleMail.properties properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mail.protocol.user</td>
<td>Specifies the protocol-specific user name that is used during authentication when you connect to a mail server. This property key overrides the mail.user property key. By default, this property is not required and no value is provided.</td>
</tr>
</tbody>
</table>

enrolepolicies.properties

The enrolepolicies.properties file provides standard and custom settings that support the functions of the provisioning policy.

Functions supported by this properties file includes:
- Specifying Java classes to process provisioning policy conflicts with join directives
- Specifying default and non-default join directive caching timeouts
- Declaring policy attributes to be ignored during policy compliance validation

A join directive is a set of rules that is used to determine how attributes are handled when a provisioning policy conflicts with another. Join directives use logical constructs to resolve conflicts. Examples include combining all policy attributes (union), with only common attributes (intersection), and resolving conflicts with Boolean AND or OR logic.

There are 12 types of join directives that you can use. Provisioning policy join directives take effect when more than one provisioning policy is defined for the same user (or group of users) for the same target service, service instance, or service type.

Custom join directives can be defined by writing a custom Java class, adding it to your class path, and then providing the fully qualified Java class name in the policy configuration GUI. If you extend or replace one of the existing join directive classes, you must add the custom property key and value to the enrolepolicies.properties file. For example if you developed a new class (com.abc.TextualEx) to replace the existing class for textual joins, the registration line is as follows:

```java
provisioning.policy.join.Textual= com.abc.TextualEx
```

Table 23 defines the properties used to configure IBM Security Identity Manager policies.

Table 23. enrolepolicies.properties properties

| Join directive classes |
Table 23. enrolepolicies.properties properties (continued)

<table>
<thead>
<tr>
<th>Property Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>provisioning.policy.join.PrecedenceSequence=com.ibm.itim.policy.join.PrecedenceSequence</td>
<td>Specifies the Java class used to process the logic of a join directive that is required to resolve a provisioning policy conflict.</td>
</tr>
<tr>
<td>provisioning.policy.join.Boolean=com.ibm.itim.policy.join.Boolean</td>
<td>Specifies the Java class used to process boolean join directives.</td>
</tr>
<tr>
<td>provisioning.policy.join.Bitwise=com.ibm.itim.policy.join.Bitwise</td>
<td>Specifies the Java class used to process bitwise join directives.</td>
</tr>
<tr>
<td>provisioning.policy.join.Numeric=com.ibm.itim.policy.join.Numeric</td>
<td>Specifies the Java class used to process numeric join directives.</td>
</tr>
<tr>
<td>provisioning.policy.join.Textual=com.ibm.itim.policy.join.Textual</td>
<td>Specifies the Java class used to process textual join directives.</td>
</tr>
<tr>
<td>provisioning.policy.join.Textual.AppendSeparator=&lt;&lt;&lt;&gt;&gt;</td>
<td>Specifies the character that is used by the textual join directive Java class to separate individual values of a multi-value attribute. Example: provisioning.policy.join.Textual.AppendSeparator=&lt;&lt;&lt;&gt;&gt;</td>
</tr>
<tr>
<td>provisioning.policy.join.Multivalued=com.ibm.itim.policy.join.Multivalued</td>
<td>Specifies the Java class used to process multivalued join directives.</td>
</tr>
</tbody>
</table>

Do not modify these property keys and values. Each property key specifies a Java class. It can be used to process the logic of a join directive that is required to resolve a provisioning policy conflict.

Append separator characters

<table>
<thead>
<tr>
<th>Property Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>provisioning.policy.join.Textual.AppendSeparator</td>
<td>Specifies the character that is used by the textual join directive Java class to separate individual values of a multi-value attribute. Example: provisioning.policy.join.Textual.AppendSeparator=&lt;&lt;&lt;&gt;&gt;</td>
</tr>
</tbody>
</table>

Join directive cache timeouts

<table>
<thead>
<tr>
<th>Property Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>provisioning.policy.join.defaultCacheTimeout</td>
<td>Specifies the timeout interval [in seconds] between refreshes of the cache that stores default join directive cache values. The default is 86400 seconds, which is 24 hours. Example (default): provisioning.policy.join.defaultCacheTimeout=86400</td>
</tr>
<tr>
<td>provisioning.policy.join.overridingCacheTimeout</td>
<td>Specifies the timeout interval [in seconds] between refreshes of the cache that stores non-default join directive values. The default is 300 seconds, which is 5 minutes. Example: provisioning.policy.join.overridingCacheTimeout=300</td>
</tr>
</tbody>
</table>

Account attributes ignored by policy compliance validation

Excluded generic attributes (default value=1):
Table 23. enrolepolicies.properties properties (continued)

| nonvalidateable.attribute.eraccountcompliance |
| nonvalidateable.attribute.eraccountstatus |
| nonvalidateable.attribute.erauthorizationowner |
| nonvalidateable.attribute.erglobalid |
| nonvalidateable.attribute.erhistoricalpassword |
| nonvalidateable.attribute.eraservice |
| nonvalidateable.attribute.objectclass |
| nonvalidateable.attribute.ownernoise183b2be |  

Excluded Windows NT attributes:

| nonvalidateable.attribute.erntpasswordexpired |
| nonvalidateable.attribute.erntuserbadpwdcount |
| nonvalidateable.attribute.erntlockedout |

Declares account attributes that are to be ignored during policy compliance validation. This exclusion list reduces overhead during compliance validation. It also reduces the risk of system failure that can be caused by attributes that cannot logically be resolved during validation.

Partition size

policy.partition.size

To analyze many persons during a policy change event without incurring transaction timeouts, you must break apart or partition the total number of affected persons. It is done, not for starting the concurrent policy analysis, but strictly to avoid waiting in a single database transaction for all persons to be processed. Creating multiple transactions or quickly partitioning the total number of users diminishes the chance of any (smaller) transactions to exceed the transaction timeout value. When a WebSphere Application Server cluster is used with IBM Security Identity Manager, it is helpful to note that partitioning operation itself is not clustered. It is done on the same WebSphere Application Server node which receives the policy change request.

Specifies the number of persons or accounts to be evaluated in each thread during high volume policy analysis. High volume policy analysis occurs when a policy change or a service enforcement level change affects a large group of persons or accounts. A larger partition size results in fewer threads. A smaller partition size results in more executed threads in parallel, which requires more memory.

Example (default):

policy.partition.size=2500

policy.message.size
Table 23. enrolepolicies.properties properties (continued)

<table>
<thead>
<tr>
<th>Specifies the number of persons that are analyzed as part of policy change within a single JMS message. Since WebSphere Application Server polled and reuses threads, the JMS mechanism queues the individual units of analysis work for all assigned WebSphere Application Server threads or message consumers. It is likely that during large policy changes that affect large numbers of people, all JMS consumer threads are busy processing policy analysis and enforcement; the queue for each thread is saturated with more messages to process.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example (default):</td>
</tr>
<tr>
<td>policy.message.size=25</td>
</tr>
</tbody>
</table>

enroleworkflow.properties

The enroleworkflow.properties file specifies the XML file mappings for system-defined workflows.

A workflow is a process that specifies the flow of operations that involve business operations and human interactions. A workflow design defines the manner in which a particular business logic is processed. The XML files specified in the enroleworkflow.properties file implement workflow designs.

The system workflow is identified by a unique type ID and an associated XML file. The XML workflow files are in the following directory:

```
ISIM_HOME\data\workflow_systemprocess
```

Do not remove or modify the default system workflow type IDs and XML file values in the enroleworkflow.properties file.

The updating of the following XML files is not supported.

Table 24 defines the properties used to configure IBM Security Identity Manager workflows.

Table 24. enroleworkflow.properties properties

<table>
<thead>
<tr>
<th>Policy enforcement workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.workflow.PS=enforcepolicyforservice.xml</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Account fulfillment for noncompliant accounts workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.workflow.EN=fulfillpolicyforaccount.xml</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service selection management workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.workflow.SA=addserviceselectionpolicy.xml</td>
</tr>
<tr>
<td>enrole.workflow.SC=changeserviceselectionpolicy.xml</td>
</tr>
<tr>
<td>enrole.workflow.SD=removeserviceselectionpolicy.xml</td>
</tr>
</tbody>
</table>

| Provisioning policy management workflow |
Table 24. enroleworkflow.properties properties (continued)

| #Add policy                        | enrole.workflow.PA=addpolicy.xml |
| #Modify policy                    | enrole.workflow.PC=changepolicy.xml |
| #Delete policy                    | enrole.workflow.PD=removepolicy.xml |
| #User BU change                   | enrole.workflow.UO=userbuchange.xml |

Reconciliation workflow

| enrole.workflow.RC=reconciliation.xml |
| enrole.workflow.HR=hrfeed.xml |

Dynamic role workflow

| #Add dynamic role                 | enrole.workflow.DA=adddynamicrole.xml |
| #Modify dynamic role              | enrole.workflow.DC=changedynamicrole.xml |
| #Delete dynamic role              | enrole.workflow.DD=removedynamicrole.xml |
| #Import Policy Enforcement       | enrole.workflow.PE=importpolicyenforcement.xml |
| #Process Lifecycle Rule          | enrole.workflow.LC=lifecyclerule.xml |

fesiextensions.properties (deprecated)

The fesiextensions.properties file (deprecated) provides support for Free EcmaScript Interpreter (FESI) JavaScript extensions before Version 5.0 of IBM Security Identity Manager. Do not author new extensions using this deprecated architecture.

The fesiextensions.properties file defines built-in and custom FESI extensions required by IBM Security Identity Manager. FESI is the Free EcmaScript Interpreter, a JavaScript interpreter written in Java. The FESI interpreter reads this properties file during IBM Security Identity Manager initialization to set extensions for required Java classes.

The FESI extensions represent regions, or hooks, in IBM Security Identity Manager where the use of JavaScript code is allowed to introduce built-in or custom business logic. FESI extensions are specified with a property key and value pair format:

```
property-key-name=value
```

The value is a fully qualified Java class file name. The property-key-name includes a standard prefix (fesi.extension), a context, and (for custom classes) an identifier name (ID) representing the fully qualified Java class file. Typically the shorter unqualified class name is used as the identifier name (ID).

```
fesi.extension.context.class-ID=fully-qualified-class-name
```

The FESI system extensions that are used by IBM Security Identity Manager include a global context and three specific contexts.
Although you must not modify the built-in system FESI extensions, you can add custom FESI extensions that might be required for any custom programs. When you add a custom FESI extension to this properties file, you must use one of the established global or specific contexts.

Indicate the fully qualified custom Java class file name as the value and provide a unique property key identifier name (ID) for the custom class. Examples:
fesi.extension.IdentityPolicy.custom-class-ID=custom-fully-qualified-class-name
fesi.extension.HostSelection.custom-class-ID=custom-fully-qualified-class-name

Table 25 defines the deprecated properties used to configure FESI extensions (on a single line).

<table>
<thead>
<tr>
<th>System FESI extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>fesi.extension.Enrole=com.ibm.itim.fesiextensions.Enrole</td>
</tr>
<tr>
<td>fesi.extension.HostSelection=com.ibm.itim.fesiextensions.ModelExtension</td>
</tr>
<tr>
<td>fesi.extension.Workflow=com.ibm.itim.workflow.fesiextensions.WorkflowExtension</td>
</tr>
</tbody>
</table>

The value for each system property key is a fully qualified Java class file that IBM Security Identity Manager provides.

Do not remove or modify information in this section.

<table>
<thead>
<tr>
<th>Custom FESI extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
</tr>
<tr>
<td>fesi.extension.enRole.custom-class-ID=custom-fully-qualified-class-name</td>
</tr>
</tbody>
</table>

You can modify the fesiextensions.properties files to include additional FESI extensions for required custom objects and methods.

The value for each custom property key is a fully qualified custom Java class file.

All property key names must be unique.

<table>
<thead>
<tr>
<th>JavaScript password access</th>
</tr>
</thead>
<tbody>
<tr>
<td>javascript.password.access.enabled</td>
</tr>
</tbody>
</table>
Table 25. fesiextensions.properties properties (deprecated) (continued)

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| `javascript.password.access.enabled` | Determines whether plaintext passwords can be accessed from Person and Account objects. Values include:  
  - `true` — Password access is enabled.  
  - `false` — Passwords cannot be accessed with javascript.  
  Example (default): `javascript.password.access.enabled=true` |

helpmappings.properties

The helpmappings.properties file allows a customer to replace the installed IBM Security Identity Manager help system with an alternative help system.

The helpmappings.properties file contains the following properties:

Table 26. helpmappings.properties properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| `url.contexthelp` | Specifies an external URL for help. The default is blank, which uses the URL of the IBM Security Identity Manager help system. The URL will also add the resolved locale based on the IBM Security Identity Manager language packs that are installed. For example, `http://www.timcustomer.com/help/en/ui_login.html`  
  Example: `url.contexthelp=www.timcustomer.com/help`  
  Clicking on the help icon (‘?’) in the IBM Security Identity Manager graphical user interface will load the html file from the key mapping (`http://www.timcustomer.com/help/customerfilename.html`). For a login page, the value of `customerfilename` might be `ui_login.html`, and the full address might be `http://www.timcustomer.com/help/ui_login.html`. |

reportingLabels.properties

This properties file is like other labels-related properties files such as labels.properties, or customLabels.properties, and holds labels that are used by Reports.

reporttabledeny.properties

By default, this property holds a list of IBM Security Identity Manager tables that are used by various IBM Security Identity Manager components to store internal or configuration data that is inappropriate for a report.

This file is used by IBM Security Identity Manager Server for Reporting Engine purposes.

The following table defines the properties that determine which information is not exposed in reports.

Table 27. reporttabledeny.properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tables</code></td>
<td></td>
</tr>
</tbody>
</table>
### reporttabledeny.properties (continued)

<table>
<thead>
<tr>
<th><strong>Holds a comma-separated list of all IBM Security Identity Manager database tables that are excluded from report production.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>If a table is part of this property, the table and its columns are not in the Report Designer; a report cannot be designed on columns of this table. A user who wants to deny a specific database table from being used by the Report Designer can choose to add the table against the <code>tables</code> property.</td>
</tr>
<tr>
<td>Example: tables=JMSState, JMSStore, entity_column, column_report, report, synchronization_history, synchronization_lock, changelog, resources_synchronizations, NextValue, ListData, AUTH_KEY, ATTR_CHANGE, ACCT_CHANGE, LCR_INPROGRESS_TABLE, WORKFLOW_CALLBACK, POLICY_ANALYSIS, POLICY_ANALYSIS_ERROR, PO_TOPIC_TABLE, PO_NOTIFICATION_TABLE, BULK_DATA_STORE, BULK_DATA_INDEX, MANUAL_SERVICE_RECON_ACCOUNTS, SCRIPT, ACTIVITY_LOCK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>allowedRestrictedColumns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allows IBM Security Identity Manager administrators to explicitly allow columns of restricted data types, to be used for designing and running custom reports. Such reports however work for IBM Security Identity Manager Administrators only. If a non-administrator attempts to run such reports, the user receives an AuthorizationException.</td>
</tr>
<tr>
<td>By default, columns of the following restricted data types are not available when you design or run custom reports: BLOB, CLOB, BINARY, VARBINARY, LONGVARBINARY and LONGVARCHAR</td>
</tr>
<tr>
<td>The value of the property is a comma-separated list of <code>&lt;TABLE_NAME&gt;.&lt;COLUMN_NAME&gt;</code>. If this property is undefined, then none of the columns of the restricted data type is available for reporting.</td>
</tr>
<tr>
<td>Example (on a single line): allowedRestrictedColumns=ACTIVITY.RESULT_DETAIL, PROCESS.RESULT_DETAIL, PROCESSLOG.NEW_DATA</td>
</tr>
</tbody>
</table>

---

### scriptframework.properties (Suggested)

For all new JavaScript extensions, use the `scriptframework.properties` file to configure script extensions and other scripting functions.

JavaScript is used in IBM Security Identity Manager to specify identity policies, provisioning policy parameters, service selection policies, placement rules for identity feeds, and orphan account adoption.

In addition, JavaScript is used in workflows to specify transition conditions, loop conditions, JavaScript activities, activity postscripts, and workflow notification. Various scripting extensions are provided by IBM Security Identity Manager to expose useful data and services to each of these scripts. In addition to these extensions, system administrators can configure IBM Security Identity Manager to load custom JavaScript extensions.

The file `scriptframework.properties` is used to configure all parts of scripting support in IBM Security Identity Manager. It includes which script extensions to use, which script interpreter to use, and other properties that relate to scripting.
The major parts of the scriptframework.properties are divided by these host components: PostOffice, ProvisioningPolicy, AccountTemplate, HostSelection, PersonPlacementRules, Workflow, Reminder, IdentityPolicy, Notification, and OrphanAdoption.

The most heavily used section of the property file is for configuring which extensions to load for each host component. To have the script framework load an extension, add a key-value line to the scriptframework.properties file that is similar to this example:

```
ITIM.extension.{Host Component}=com.ibm.itim.class_name
```

where ITIM.extension.{Host Component} is the key and com.ibm.itim.class_name is the value. The value of {Host Component} can be any of the previously listed components. If you want to load more than a single extension for a host component, you can add a suffix to host component, such as:

```
ITIM.extension.{Host Component}.suffix=com.ibm.itim.class_name
```

The only rule is that each key must be unique in the file.

The scriptframework.properties file comes pre-configured to load the extensions necessary to use IBM Security Identity Manager with its default scripts. Do not remove any lines in scriptframework.properties because removal might cause IBM Security Identity Manager to stop functioning properly.

The next section of the scriptframework.properties file configures which script interpreter to use for each host component. IBM Security Identity Manager currently supports two different script interpreters, the IBM JSEngine and the FESI JavaScript Interpreter.

To configure which interpreter to use for each host component, there is a line in the file that looks like:

```
ITIM.interpreter.{Host Component}={Engine}
```

The value of {Host Component} can be any of the previously listed components. The value of {Engine} can be either IBMJS or FESI. The {Engine} variable is not case-sensitive, so typing fesi works as well as typing FESI. IBMJS is the default scripting engine, so any value for {Engine} other than IBMJS or FESI, or no value, uses the IBMJS engine. The FESI engine is deprecated. Use it only if you upgraded from IBM Security Identity Manager Version 4.6 or earlier and have custom FESI extensions.

The next section in the configuration file enables configuring custom JavaScript wrappers. For security reasons, IBM Security Identity Manager does not expose all objects to the scripting environment. Instead, most objects are wrapped in a more restrictive wrapper class that exposes only certain methods. IBM Security Identity Manager has a default wrapper configuration that you can override or extend in this section. This feature is for an advanced user; in most cases do not use it. For more details on how to configure custom wrappers, see the comments in the scriptframework.properties file.

In the next section, you can configure direct Java access from scripts run by the IBM JSEngine interpreter. Direct Java access is powerful, but scripts can bypass some of the security built into the script framework. Consider carefully before you do so. See the comments in the scriptframework.properties file for more information about how to enable direct Java access.
The final section of the configuration file configures specific properties that might be useful. Each specific property is explained in comments in the `scriptframework.properties` file, including default and allowed values.

**SelfServiceHelp.properties**

The `SelfServiceHelp.properties` file can be used to redirect help to a custom location for customers who want to have their own help content for the self-service user interface.

Table 28 defines the properties used to redirect help to a custom location.

**Table 28. SelfServiceHelp properties**

<table>
<thead>
<tr>
<th>IBM Security Identity Manager SelfServiceHelp settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>helpBaseUrl</strong></td>
</tr>
<tr>
<td>Specifies the base url to send help requests to. A blank value indicates that help goes to the URL for Self Service application help.</td>
</tr>
<tr>
<td>Valid values include the URL of the Self Service application help.</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td><code>helpBaseUrl=http://myserver:80</code></td>
</tr>
</tbody>
</table>

Help Id mappings include:

- `helpId = relative page URL`

The help mappings section maps ids from specific pages to a relative URL sent to the help server.

For example:

- `helpBaseUrl=http://myserver:80`
- `locale = en_US`
- `loginId/relativeURL = login_help_url=ui/ui_eui_login.html`

Locale is determined by resolving the `SelfServiceScreenText.properties` resource bundle for the current logged in user and with the associated locale.

**SelfServiceHomePage.properties**

The `SelfServiceHomePage.properties` file is used to configure the sections of the initially installed home page for the self-service user interface. You can add or remove tasks, and update icon URLs and labels of the home page from this file.

The file has these types of entries:

- **Sections=**ActionNeeded, Password, sectionConfigName ...
  - Defines the section configuration names in the order in which they are displayed.
- **Section definition**
  - Defines the label keys, icons, and other objects for the home page section.
- **Task definitions**
  - Defines the NLS key and link for the URL, the NLS key for the task description, and other attributes that enable displaying the task.
For more information about these properties, see documentation in the properties file.

**SelfServiceScreenText.properties**

The `SelfServiceScreenText.properties` file is a resource bundle containing the labels for the self-service user interface.

Versions of the file might be available for the installed languages. For example: `SelfServiceScreenText_en.properties` and `SelfServiceScreenText_es.properties`, which are editable by users.

**SelfServiceUI.properties**

The `SelfServiceUI.properties` file controls miscellaneous properties of the self-service user interface.

Table 29 defines the properties used to configure the self-service user interface.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.ui.pageSize</code></td>
<td>Specifies the page size for displaying lists.</td>
<td><code>enrole.ui.pageSize=10</code></td>
</tr>
<tr>
<td><code>enrole.ui.pageLinkMax</code></td>
<td>Specifies the number of page links to be shown for multi-page result sets.</td>
<td><code>enrole.ui.pageLinkMax=100</code></td>
</tr>
<tr>
<td><code>enrole.ui.maxSearchResults</code></td>
<td>Specifies the maximum number of items returned from a search. The results that are returned can be less than, but not larger than the values specified in <code>ui.properties</code>.</td>
<td><code>enrole.ui.maxSearchResults=1000</code></td>
</tr>
<tr>
<td><code>enrole.ui.maxSearchResults.users</code></td>
<td>Specifies the maximum displayable search results for the task Delegate Activities - Search for User.</td>
<td><code>enrole.ui.maxSearchResults.users=100</code></td>
</tr>
<tr>
<td><code>enrole.ui.maxNrOfIteration</code></td>
<td>Specifies the maximum number of wait iterations for RequestInfo status.</td>
<td><code>enrole.ui.maxNrOfIteration=20</code></td>
</tr>
<tr>
<td><code>enrole.ui.waitTime</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>enrole.ui.waitTime</td>
<td>Specifies the time to wait until the request is asked for next status.</td>
<td>enrole.ui.waitTime=3000</td>
</tr>
<tr>
<td></td>
<td>The product of (maxNrOfIteration * waitTime) is a maximum of 60 seconds.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The value is interpreted in milliseconds.</td>
<td></td>
</tr>
<tr>
<td>enrole.ui.logoffURL</td>
<td>Specifies the URL to forward the browser to when the user logs off.</td>
<td>enrole.ui.logoffURL=myLogoffURL</td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>enrole.ui.timeoutURL</td>
<td>Specifies the URL to which forward the browser on timeout.</td>
<td>enrole.ui.timeoutURL=myTimeoutURL</td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>ui.layout.showBanner</td>
<td>Specifies a change to the values of ui.layout properties to show or hide</td>
<td>ui.layout.showBanner=true</td>
</tr>
<tr>
<td></td>
<td>the banner of the self-service user interface.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>ui.layout.showFooter</td>
<td>Specifies a change to the values of ui.layout properties to show or hide</td>
<td>ui.layout.showFooter=true</td>
</tr>
<tr>
<td></td>
<td>the footer of the self-service user interface.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>ui.layout.showToolbar</td>
<td>Specifies a change to the values of ui.layout properties to show or hide</td>
<td>ui.layout.showToolbar=true</td>
</tr>
<tr>
<td></td>
<td>the toolbar of the self-service user interface.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>ui.layout.showNav</td>
<td>Specifies a change to the values of ui.layout properties to show or hide</td>
<td>ui.layout.showNav=false</td>
</tr>
<tr>
<td></td>
<td>the page navigation of the self-service user interface.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>ui.usersearch.attr.cn</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 29. SelfServiceUI.properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ui.usersearch.attr.cn</code></td>
<td>Specifies the attribute that is listed in the <code>searchBy</code> field for a user search. The attribute is prefixed with <code>ui.usersearch.attr</code>. For more information about mapping and syntax, see the documentation in the SelfServiceUI.properties file.</td>
<td><code>ui.usersearch.attr.cn=cn</code></td>
</tr>
<tr>
<td><code>ui.usersearch.attr.sn</code></td>
<td>Specifies the attribute that is listed in the <code>searchBy</code> field for a user search. The attribute is prefixed with <code>ui.usersearch.attr</code>. For more information about mapping and syntax, see the documentation in the SelfServiceUI.properties file.</td>
<td><code>ui.usersearch.attr.sn=sn</code></td>
</tr>
<tr>
<td><code>ui.usersearch.attr.telephonenumber</code></td>
<td>Specifies the attribute that is listed in the <code>searchBy</code> field for a user search. The attribute is prefixed with <code>ui.usersearch.attr</code>. For more information about mapping and syntax, see the documentation in the SelfServiceUI.properties file.</td>
<td><code>ui.usersearch.attr.telephonenumber=telephonenumber</code></td>
</tr>
<tr>
<td><code>ui.usersearch.attr.mail</code></td>
<td>Specifies the attribute that is listed in the <code>searchBy</code> field for a user search. The attribute is prefixed with <code>ui.usersearch.attr</code>. For more information about mapping and syntax, see the documentation in the SelfServiceUI.properties file.</td>
<td><code>ui.usersearch.attr.mail=mail</code></td>
</tr>
</tbody>
</table>
| `ui.view.accounts.expandedbydefault` | Specifies whether the accounts affected twistie state on the change password page are expanded or collapsed (`true|false`) by default. Valid values are:  
  - `true` – Expand the accounts affected twistie state on the change password page by default  
  - `false` – Do not expand the accounts affected twistie state on the change password page by default | `ui.view.accounts.expandedbydefault=false` |
| `ui.select.all.accounts`          | Specifies whether all the accounts under the account twistie are to be selected by default. Valid values are:  
  - `all` – To select all the accounts under the account twistie  
  - `none` – To select none of the accounts under the account twistie  
  - `default` – To retain the default behavior | `ui.select.all.accounts=default` |
The `ui.properties` file specifies attributes that affect the operation and display of the IBM Security Identity Manager graphical user interface.

Table 30 defines the properties that are used to configure the IBM Security Identity Manager graphical user interface.

<table>
<thead>
<tr>
<th>IBM Security Identity Manager GUI configuration settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>enrole.ui.customerLogo.image</strong></td>
</tr>
</tbody>
</table>
| Specifies the file name of the graphic that is displayed on the right side of the IBM Security Identity Manager title banner. The graphic is usually a company logo. For display over the web in a browser, the format of the file must be type that the browser supports. The actual graphics file must be stored in the following location: WebSphere Application Server: WebSphere/AppServer/installedApps/domain-name/ITIM.ear/itim_console.war/html/images/ You can also specify a path under the console webapp, /itim/console/custom/banner.gif or specify the full URL as http://yourhost.com/banner.gif. Example: 
enrole.ui.customerLogo.image=ibm_banner.gif |
| **enrole.ui.customerLogo.url**                          |
| Specifies the URL link that is activated when you click the custom graphic image (company logo) on the right side of the IBM Security Identity Manager banner. Example: 
enrole.ui.customerLogo.url=www.ibm.com |
| **enrole.ui.pageSize**                                  |
| Specifies the number of list items that is initially displayed on the screen. If there are more items in the list, links are at the bottom of the list view that activate continuations of the list. For example, Page 2, Page 3, Page 4. Example: 
enrole.ui.pageSize=50 |
| **enrole.ui.maxSearchResults**                          |
| Specifies the maximum number of items that are returned for a search. This property limits the number of items that are returned when a search is done on the directory server. The evaluation of the ACIs is done later on these returned items. The number of items in the directory server is greater than the value specified for this property. So, the number of items that are displayed on the IBM Security Identity Manager Console might be less than the value specified. The value for this property can control possible system performance degradation when a large return of items is encountered. If you modify the value for this property, you must restart the application server. Example: 
enrole.ui.maxSearchResults=1000 |
| **ui.banner.showForLogin**                              |
### Table 30. `ui.properties` properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Values</th>
<th>Example (default):</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ui.banner.showForLogin</code></td>
<td>Specifies whether to show the console banner on the login page, rather than the default login banner. Any customization to the console banner is also on the login page when this property is in effect.</td>
<td><code>yes</code> (Show the console banner in the login page.)&lt;br&gt;<code>no</code> (Show the default login banner. An empty value assumes no.)</td>
<td><code>ui.banner.showForLogin=no</code></td>
</tr>
<tr>
<td><code>ui.footer.URL</code></td>
<td>Specifies the URL for the IBM Security Identity Manager Console. Specify either the full address (<code>http://yourhost.com/footer.html</code>) or an address from the IBM Security Identity Manager web server (<code>/itim/console/custom/footer.html</code>). A blank value uses the default address of the IBM Security Identity Manager footer.</td>
<td><code>ui.footer.URL=http://itim99.mylab.raleigh.ibm.com:9080/itim/console/main</code></td>
<td></td>
</tr>
<tr>
<td><code>ui.footer.height</code></td>
<td>Specifies the height in pixels of the footer on the IBM Security Identity Manager Console.</td>
<td><code>ui.footer.height=50</code></td>
<td></td>
</tr>
<tr>
<td><code>ui.footer.isVisible</code></td>
<td>Shows or hides the footer of the IBM Security Identity Manager Console.</td>
<td><code>yes</code> (or blank) (Shows the footer.)&lt;br&gt;<code>no</code> (Hides the footer.)</td>
<td><code>ui.footer.isVisible=yes</code></td>
</tr>
<tr>
<td><code>ui.banner.height</code></td>
<td>Specifies the height in pixels of the banner on the IBM Security Identity Manager Console.</td>
<td><code>ui.banner.height=48</code></td>
<td></td>
</tr>
<tr>
<td><code>ui.homepage.path</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Property Name</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>ui.properties</code></td>
<td><strong>IBM Security Identity Manager Console home page location.</strong> Specify a relative path from the IBM Security Identity Manager Console context root (<code>/itim/console</code>). For example, if the full path to the home page was <code>http://yourhost:9080/itim/console/custom/home.html</code>, then the following value is <code>ui.homepage.path=custom/home.html</code>. The custom home page must be in the IBM Security Identity Manager web application. For example: <code>path/ITIM.ear/itim_console.war/custom/home.html</code>). A blank value uses the default address of the IBM Security Identity Manager home page. Example: <code>ui.homepage.path=custom/home.html</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>ui.titlebar.text</code></td>
<td><strong>Specifies the text in the title bar of the browser for the IBM Security Identity Manager Console.</strong> A blank value uses the default name of the IBM Security Identity Manager product. Example: <code>ui.titlebar.text=Our Home Page</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>ui.userManagement.includeAccounts</code></td>
<td><strong>Specifies the default behavior for including accounts when you suspend, restore, or delete users.</strong> Valid values are as follows: <code>true</code> Accounts are included. <code>false</code> Accounts are excluded. Example (default): <code>ui.userManagement.includeAccounts=true</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>ui.userManagement.search.attributes</code></td>
<td><strong>Adds a search attribute to the default list for the Manage Users page in the IBM Security Identity Manager Console. Provide one or more attribute names in the <code>ui.userManagement.search.attributes</code> property value that is separated by a comma. Make sure to provide valid and non-repetitive attributes. Do not specify attributes that cannot be searched by using plain text.</strong> For example, audio, photo, and other similar items. Example: <code>ui.userManagement.search.attributes=homepostaladdress,employeenumber</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>ui.challengeResponse.showAnswers</code></td>
<td><strong>The property adds user attributes that display in the Search By list on the Manage Users page for the person search filter.</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Chapter 15. Supplemental property files  223
<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ui.properties</td>
<td>Specifies whether the answers to challenge response questions is treated as</td>
</tr>
<tr>
<td></td>
<td>passwords or as clear text in the IBM Security Identity Manager Console of the</td>
</tr>
<tr>
<td></td>
<td>following pages:</td>
</tr>
<tr>
<td></td>
<td>· <strong>Forgot Password</strong> page</td>
</tr>
<tr>
<td></td>
<td>· Challenge response question and answer definition page</td>
</tr>
<tr>
<td></td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td>true  Answers to challenge response questions is clear text.</td>
</tr>
<tr>
<td></td>
<td>false Answers to challenge response questions is treated as passwords.</td>
</tr>
<tr>
<td></td>
<td>Example (default):</td>
</tr>
<tr>
<td></td>
<td>ui.challengeResponse.showAnswers=true</td>
</tr>
<tr>
<td>ui.challengeResponse.bypassChallengeResponse</td>
<td>Specifies whether the challenge response questions can be bypassed when the user first logs on to the IBM Security Identity Manager Console or the self service web user interface. Valid values:</td>
</tr>
<tr>
<td></td>
<td>true When true, the user can cancel and not answer the challenge questions.</td>
</tr>
<tr>
<td></td>
<td>false When false, the user cannot cancel. The user is forced to respond to the challenge questions.</td>
</tr>
<tr>
<td></td>
<td>Default value: true</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>ui.challengeResponse.bypassChallengeResponse=true</td>
</tr>
<tr>
<td>ui.viewAllRequests.loadDefaultQueryResult</td>
<td>Specifies whether the <strong>View All Requests</strong> page loads the default query result.</td>
</tr>
<tr>
<td></td>
<td>true Loads the <strong>View All Requests</strong> page with default query result.</td>
</tr>
<tr>
<td></td>
<td>false Does not load the <strong>View All Requests</strong> page with default query result.</td>
</tr>
<tr>
<td></td>
<td>Default value: false</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>ui.viewAllRequests.loadDefaultQueryResult=false</td>
</tr>
<tr>
<td>ui.allowLaunchingNewTaskWithoutWarningForActiveTask</td>
<td>Specifies whether to start selected task or not, if the same task is already active in the IBM Security Identity Manager Console. The examples of the tasks are as follows: Create Service, Change Service, Create User, Change User.</td>
</tr>
<tr>
<td></td>
<td>true When you try to start an already active task, the existing task is closed. Starts the new task without displaying any warning message.</td>
</tr>
<tr>
<td></td>
<td>false When you try to start an already active task, a warning message is displayed. Does not start the new task.</td>
</tr>
<tr>
<td></td>
<td>Default value: false</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>ui.allowLaunchingNewTaskWithoutWarningForActiveTask=false</td>
</tr>
<tr>
<td>ui.policyManagement.manageProvisioningPolicies.create.defaultMemberType</td>
<td></td>
</tr>
</tbody>
</table>
Table 30. `ui.properties` properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Allowed Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ui.properties</code> properties</td>
<td>Controls default selection of policy membership. This property allows default member type to be selected while you create a provisioning policy. Allowed values are as follows:</td>
<td></td>
</tr>
<tr>
<td><code>users</code></td>
<td>All users in the organization.</td>
<td></td>
</tr>
<tr>
<td><code>roles</code></td>
<td>Roles that are specified later.</td>
<td></td>
</tr>
<tr>
<td><code>others</code></td>
<td>All other users who are not granted to the entitlements that are defined by this provisioning policy by way of other policies.</td>
<td></td>
</tr>
<tr>
<td>Default value: <code>users</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td><code>ui.policyManagement.manageProvisioningPolicies.create.defaultMemberType=users</code></td>
</tr>
<tr>
<td><code>ui.manageServices.reconcileNow.defaultSelectQuery</code></td>
<td>Specifies the default reconciliation query option. Allowed values are as follows:</td>
<td></td>
</tr>
<tr>
<td><code>none</code></td>
<td>None.</td>
<td></td>
</tr>
<tr>
<td><code>use_query</code></td>
<td>Use query from existing schedule.</td>
<td></td>
</tr>
<tr>
<td><code>define_query</code></td>
<td>Define query.</td>
<td></td>
</tr>
<tr>
<td>Default value: <code>none</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td><code>ui.manageServices.reconcileNow.defaultSelectQuery=none</code></td>
</tr>
<tr>
<td><code>ui.passwordManagement.defaultSelection.typePassword</code></td>
<td>Specifies <strong>Allow me to type a password</strong> as default over the current <strong>Generate a password for me</strong> option. Allowed values are as follows:</td>
<td></td>
</tr>
<tr>
<td><code>true</code></td>
<td>Selects the <strong>Allow me to type a password</strong> option and additionally none of the accounts get selected by default.</td>
<td></td>
</tr>
<tr>
<td><code>false</code></td>
<td>Selects the <strong>Generate a password for me</strong> option if this property is set to false or not present.</td>
<td></td>
</tr>
<tr>
<td>Default value: <code>false</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td><code>ui.passwordManagement.defaultSelection.typePassword=false</code></td>
</tr>
<tr>
<td><code>ui.advancedUserSearch.AllTypes.defaultSearchAttribute.names</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>ui.advancedUserSearch.AllTypes.defaultSearchAttribute.labels</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 30. *ui.properties* properties (continued)

<table>
<thead>
<tr>
<th>Table 30. <em>ui.properties</em> properties (continued)</th>
</tr>
</thead>
<tbody>
<tr>
<td>When you select <strong>User type</strong> as <strong>All types</strong> in the <strong>Select User Type</strong> page, the properties add the default search attributes and its labels on the <strong>Advanced Search</strong> page for users in the IBM Security Identity Manager Console. If the <code>ui.advancedUserSearch.AllTypes.defaultSearchAttribute.names</code> property is removed or if no value is specified, then IBM Security Identity Manager does not display any default search attribute field.</td>
</tr>
<tr>
<td>Provide one or more attribute names in the <code>ui.advancedUserSearch.AllTypes.defaultSearchAttribute.names</code> property value, and corresponding attribute labels in the <code>ui.advancedUserSearch.AllTypes.defaultSearchAttribute.labels</code> property value.</td>
</tr>
<tr>
<td>Make sure to provide valid, non-repetitive, and comma-separated values. Do not specify attributes that cannot be searched by using plain text. For example, audio, photo, and other similar items.</td>
</tr>
<tr>
<td><strong>Example</strong> (default): <code>ui.advancedUserSearch.AllTypes.defaultSearchAttribute.names=cn</code> <code>ui.advancedUserSearch.AllTypes.defaultSearchAttribute.labels=$cn</code></td>
</tr>
<tr>
<td>The property adds the default search attributes and its labels on the <strong>Advanced Search</strong> page for users when you select <strong>User type</strong> as <strong>All types</strong> in the <strong>Select User Type</strong> page.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>WfDesigner and FormDesigner applet properties</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.build.version</code> <code>enrole.java.plugin</code> <code>enrole.java.plugin.classid</code> <code>enrole.java.pluginspage</code> <code>enrole.java.plugin.jpi-version</code> <code>enrole.java.plugin.version</code> <code>enrole.java.entWflowHeightIE</code> <code>enrole.java.entWflowWidthIE</code> <code>enrole.java.entWflowHeightMZ</code> <code>enrole.java.entWflowWidthMZ</code> <code>enrole.java.opfWflowHeightIE</code> <code>enrole.java.opfWflowWidthIE</code> <code>enrole.java.opfWflowHeightMZ</code> <code>enrole.java.opfWflowWidthMZ</code> <code>enrole.java.joinDirHeightIE</code> <code>enrole.java.joinDirWidthIE</code> <code>enrole.java.joinDirHeightMZ</code> <code>enrole.java.joinDirWidthMZ</code> <code>enrole.java.formDesignHeightIE</code> <code>enrole.java.formDesignWidthIE</code> <code>enrole.java.formDesignHeightMZ</code> <code>enrole.java.formDesignWidthMZ</code> <code>express.java.formDesignHeightIE</code> <code>express.java.formDesignWidthIE</code> <code>express.java.formDesignHeightMZ</code> <code>express.java.formDesignWidthMZ</code> <code>#enrole.ui.logoffURL (default is commented out)</code> <code>#enrole.ui.timeoutURL (default is commented out)</code></td>
</tr>
<tr>
<td>You must not modify or remove any information for these properties in the property file.</td>
</tr>
<tr>
<td>These property key and value pairs provide the necessary Java applet support required by the web browser that runs the IBM Security Identity Manager Console.</td>
</tr>
<tr>
<td><strong>Report menu properties</strong></td>
</tr>
<tr>
<td>---------------------------</td>
</tr>
<tr>
<td><strong>enrole.ui.report.maxRecordsInReport</strong></td>
</tr>
<tr>
<td>Displays the number of records that can be displayed in a PDF report without encountering an “Out of Memory” error. The number does not ensure that PDF report generation is successful. If the report contains more records than specified by this property, PDF report generation is not attempted.</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td><code>enrole.ui.report.maxRecordsInReport=5000</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Enable or disable WebSEAL single sign-on (SSO)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>enrole.ui.ssoEnabled</strong></td>
</tr>
<tr>
<td>The property key and value pairs do not pertain to the IBM Security Identity Manager Console.</td>
</tr>
<tr>
<td>Enable or disables WebSEAL single sign-on.</td>
</tr>
<tr>
<td>More configuration is required for WebSEAL single sign-on. Valid values are as follows:</td>
</tr>
<tr>
<td><strong>true</strong> WebSEAL single sign-on is enabled.</td>
</tr>
<tr>
<td><strong>false</strong> WebSEAL single sign-on is disabled.</td>
</tr>
<tr>
<td>Example (default):</td>
</tr>
<tr>
<td><code>enrole.ui.ssoEnabled=false</code></td>
</tr>
</tbody>
</table>

| **enrole.ui.ssoEncoding** |
| Specifies the encoding that is used to decode user credentials with WebSEAL single sign-on. |
| Example (default): |
| `enrole.ui.ssoEncoding=UTF-8` |

<table>
<thead>
<tr>
<th><strong>Refresh properties</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>enrole.ui.httpRefreshSecs</strong></td>
</tr>
<tr>
<td>Defines, in seconds, the refresh rate for pages within the IBM Security Identity Manager Console. This property is used during policy previews.</td>
</tr>
<tr>
<td>Example (default):</td>
</tr>
<tr>
<td><code>enrole.ui.httpRefreshSecs=10</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Search class mapping for ObjectProfileCategory</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>The property key and value pairs do not pertain to the IBM Security Identity Manager Console and must not be modified or removed.</td>
</tr>
</tbody>
</table>
Chapter 16. System property configuration in enRole.properties

This section provides detailed information about the property keys and values contained in the ISIM_HOME\data\enRole.properties system configuration file.

The enRole.properties system configuration file contains many of the properties used to configure IBM Security Identity Manager. The file properties control the program functions and enable user customization of special features.

Properties files

Java properties files define attributes that allow customizing and control of the Java software. Standard system properties files and custom properties files are used to configure user preferences and user customization.

A Java properties file defines the values of named resources. It can specify program options such as database access information, environment settings, and special features and functions.

A properties file defines named resources with a property key and value pair format:

property-key-name=value

The property-key-name is an identifier for the resource. The value is typically the name of the actual Java object. It provides the resource or a String representing the value of the property key, such as database.name=itimdb. The statement syntax allows spaces before and after the equal (=) sign. It can span multiple lines if you place a line continuation character \ (a backslash) at the end of the line. For more information about statement syntax, see the Java language references.

WebSphere Application Server properties

WebSphere Application Server properties define values that are specific to integrating IBM Security Identity Manager with the WebSphere Application Server.

Table 31 lists these WebSphere Application Server properties.

Table 31. WebSphere application server properties

<table>
<thead>
<tr>
<th>Platform Context Factory Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.platform.contextFactory</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Do not modify this property key and value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies the Java class for the platform context factory that</td>
</tr>
<tr>
<td>defines the integration point for IBM Security Identity Manager</td>
</tr>
<tr>
<td>with the WebSphere Application Server.</td>
</tr>
<tr>
<td>Example (default, entered as a single line):</td>
</tr>
<tr>
<td>WebSpherePlatformContextFactory</td>
</tr>
</tbody>
</table>
### Table 31. WebSphere application server properties (continued)

<table>
<thead>
<tr>
<th>Application server</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.appServer.contextFactory</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td>Specifies the Java class that determines which JNDI factory to use with the WebSphere Application Server.</td>
</tr>
<tr>
<td>Example (default):</td>
</tr>
<tr>
<td>enrole.appServer.contextFactory=com.ibm.websphere.naming.WsnInitialContextFactory</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enrole.appServer.url</th>
</tr>
</thead>
<tbody>
<tr>
<td>This property key and value can be changed only by a qualified administrator.</td>
</tr>
<tr>
<td>Specifies the location of the application server naming service. This value is obtained during IBM Security Identity Manager installation.</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td>enrole.appServer.url=iiop://localhost:2809</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enrole.appServer.usertransaction.jndiname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td>Specifies the JNDI name of the JTA (Java Transaction API) User Transaction object.</td>
</tr>
<tr>
<td>Example (default):</td>
</tr>
<tr>
<td>enrole.appServer.usertransaction.jndiname=jta/usertransaction</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enrole.appServer.systemUser</th>
</tr>
</thead>
<tbody>
<tr>
<td>This property key and value can be changed only by a qualified administrator. Modify with the runConfig utility only.</td>
</tr>
<tr>
<td>Specifies the name of the administrator for the WebSphere Application Server when security is enabled. In a WebSphere Application Server environment, this value is required only when global security is enabled. The value is not set if security is not enabled.</td>
</tr>
<tr>
<td>The value is used to start, stop, and configure the IBM Security Identity Manager Server. The value is also used by IBM Security Identity Manager installation and configuration routines to authenticate to the WebSphere Application Server.</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td>enrole.appServer.systemUser=system</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enrole.appServer.systemUser.credentials</th>
</tr>
</thead>
<tbody>
<tr>
<td>This property key and value can be changed only by a qualified administrator. Modify with the runConfig utility only. This value is stored in an encrypted format that depends on the option selected with the runConfig utility.</td>
</tr>
<tr>
<td>Specifies the password for the systemUser.</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td>enrole.appServer.systemUser.credentials=password</td>
</tr>
<tr>
<td>Property Key and Value</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>enrole.appServer.ejbuser.principal</td>
</tr>
<tr>
<td>enrole.appServer.ejbuser.credentials</td>
</tr>
<tr>
<td>enrole.appServer.realm</td>
</tr>
<tr>
<td>enrole.appServer.registry</td>
</tr>
<tr>
<td>enrole.appServer.security.domain</td>
</tr>
<tr>
<td>enrole.appServer.alwayssetisolevelrc</td>
</tr>
</tbody>
</table>
Table 31. WebSphere application server properties (continued)

<table>
<thead>
<tr>
<th>Property Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td>This property specifies that IBM Security Identity Manager must always set the transaction isolation level to Read-Committed when it acquires database connections.</td>
</tr>
<tr>
<td>Because the WebSphere Application Server has internal support for setting the isolation level, this property must be set to false.</td>
</tr>
<tr>
<td>Example (default): enrole.appServer.alwayssetisolevelrc=false</td>
</tr>
</tbody>
</table>

Login helper

enrole.appServer.loginHelper.class

<table>
<thead>
<tr>
<th>Property Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td>Specifies the Java class that is used to log each thread in to J2EE Security.</td>
</tr>
<tr>
<td>Example (default): enrole.appServer.loginHelper.class=com.ibm.itim.util.was.WAS40LoginHelper</td>
</tr>
</tbody>
</table>

Application server servlet path separator

enrole.servlet.path.separator

<table>
<thead>
<tr>
<th>Property Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td>Specifies the separator character used to specify path names to required resources.</td>
</tr>
<tr>
<td>Example (default): enrole.servlet.path.separator=.</td>
</tr>
</tbody>
</table>

Event notification system login

SystemLoginContextFactory

<table>
<thead>
<tr>
<th>Property Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td>Specifies the Java factory class for event notification system login appropriate for WebSphere Application Server.</td>
</tr>
<tr>
<td>Example (default, entered as a single line): SystemLoginContextFactory=com.ibm.itim.remoteservices.provider.itim.websphere.WSSystemLogonContextFactory</td>
</tr>
</tbody>
</table>

Remote services properties

The enrole.remoteservices.assemblyline.encodeusingUTF8 property is referred whenever IBM Security Identity Manager sends the assembly line to IBM Tivoli Directory Integrator dispatcher before running any operation. Use the UTF-8 encoding when the assembly line contains special characters such as German umlaut characters.
The value of the `enrole.remoteservices.assemblyline.encodeusingUTF8` property determines whether the assembly line sent to IBM Tivoli Directory Integrator is encoded with the UTF-8 format or not.

**Table 32. Remote services properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.remoteservices.assemblyline.encodeusingUTF8</code></td>
<td>Do not change this property key and value unless you are a qualified administrator. Specifies whether the UTF-8 encoding is used or not. Values include:</td>
</tr>
<tr>
<td></td>
<td>• <code>true</code> – Only the UTF-8 encoding is used.</td>
</tr>
<tr>
<td></td>
<td>• <code>false</code> – The platform default encoding is used.</td>
</tr>
<tr>
<td></td>
<td>Example (default): <code>enrole.remoteservices.assemblyline.encodeusingUTF8=false</code></td>
</tr>
</tbody>
</table>

**Web services properties**

The web services properties define the properties that are used by IBM Security Identity Manager to manage the web services API.

**Table 33** determines the web services properties.

**Table 33. Web services properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.webServices.version</code></td>
<td>Do not change this property key. Specifies the web services version. The value is returned by the <code>WSUnAuthService.getWebServicesVersion</code> web services API. Values include the version of the web services. Example (default): <code>enrole.webServices.version=1.0</code></td>
</tr>
<tr>
<td><code>enrole.webseal.ltpa.cookie.name</code></td>
<td>Do not change this property key and value unless you are a qualified administrator. Specifies the property to identify the name of the HTTP header, which carries the LTPA token. Use this property in SSO mode only. The default value is <code>LtpaToken2</code>. Do not change this property unless the HTTP header name that carries the LTPA token is other than the default specified. Example (default): <code>enrole.webseal.ltpa.cookie.name=LtpaToken2</code></td>
</tr>
<tr>
<td><code>enrole.webServices.session.cache.maxRetry</code></td>
<td></td>
</tr>
</tbody>
</table>
Table 33. Web services properties (continued)

<table>
<thead>
<tr>
<th>Property Key</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.webServices.session.cache.maxRetry</code></td>
<td>Specifies the maximum number of retries for a session cache.</td>
<td>5</td>
</tr>
<tr>
<td><code>enrole.webServices.session.mgmt.clientSide</code></td>
<td>Specifies whether the session management is client side or server side.</td>
<td>true</td>
</tr>
<tr>
<td><code>authTokenTimeout</code></td>
<td>Specifies the time in hours for how long a session can be valid.</td>
<td>48</td>
</tr>
<tr>
<td><code>sessionInactivityTime</code></td>
<td>Specifies the time in minutes for how long an unused session is active.</td>
<td>15</td>
</tr>
</tbody>
</table>

Application server properties

Application server properties define properties that are specific to the application server, such as a user-selected locale.
Table 34 defines the properties that are specific to the application server.

**Table 34. Application server properties**

<table>
<thead>
<tr>
<th>User-selected locale</th>
</tr>
</thead>
<tbody>
<tr>
<td>locale</td>
</tr>
</tbody>
</table>

  - Specifies the locale setting for the IBM Security Identity Manager environment.
  - Example (default):
    - locale=en

<table>
<thead>
<tr>
<th>Context factory name</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.appServer.name</td>
</tr>
</tbody>
</table>

  - Specifies the unique name of the application server.
  - In a cluster environment, it is important that this name is unique for each member in the cluster.
  - Example (default):
    - enrole.appServer.name=myserver

<table>
<thead>
<tr>
<th>enrole.password.database.encrypted</th>
</tr>
</thead>
</table>

  - Use the `runConfig` utility to modify this property.
  - Specifies whether the password for the database connection (specified by the `database.db.password` property in the `enroleDatabase.properties` file) is encrypted. Valid values are:
    - `true` – Password is encrypted.
    - `false` – Password is not encrypted.
  - Example (default):
    - enrole.password.database.encrypted=true

<table>
<thead>
<tr>
<th>enrole.password.ldap.encrypted</th>
</tr>
</thead>
</table>

  - Use the `runConfig` utility to modify this property.
  - Specifies whether the LDAP password (specified by the `java.naming.security.credentials` property in the `enRoleLDAPConnection.properties` file) is encrypted. Valid values are:
    - `true` – Password is encrypted.
    - `false` – Password is not encrypted.
  - Example (default):
    - enrole.password.ldap.encrypted=true

<table>
<thead>
<tr>
<th>enrole.password.appServer.encrypted</th>
</tr>
</thead>
</table>

  - Use the `runConfig` utility to modify this property.
  - Specifies whether the password (specified by the `java.naming.security.credentials` property in the `enRoleLDAPConnection.properties` file) is encrypted. Valid values are:
    - `true` – Password is encrypted.
    - `false` – Password is not encrypted.
Table 34. Application server properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| enrole.appServer.ejbuser.credentials.encrypted | Indicates whether the application server password is encrypted. Valid values are:
  - true - Password is encrypted.
  - false - Password is not encrypted. Example (default):
    enrole.password.appServer.encrypted=true |

Organization properties

Organization properties define the organization name that is used by the directory server.

Table 35 defines the properties for the organization name that is used by the directory server.

Table 35. Organization properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| enrole.defaulttenant.id          | Specifies the short format of the organization name that is used by the directory server. This value is specified during installation of IBM Security Identity Manager or by running the ldapConfig utility. Example (default):
  enrole.defaulttenant.id=org |
  In LDAP, this value is expressed as:
  ou=org |
| enrole.organization.name        | Specifies the long format of the organization name that is used by the directory server. This value is specified during installation of IBM Security Identity Manager or by running the ldapConfig utility. Example (default):
  enrole.organization.name=Organization |
**LDAP server properties**

LDAP server properties define the properties that are used by the directory server in which IBM Security Identity Manager stores data.

Table 36 defines the properties that are used the directory server.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.ldapserver.root</code></td>
<td>Specifies the top-level entry node of the directory server data structure (<code>dc=domain control</code>). Use the <code>ldapConfig</code> utility to modify this value.</td>
</tr>
<tr>
<td></td>
<td>This value is specified during installation of IBM Security Identity Manager.</td>
</tr>
<tr>
<td></td>
<td>Example (default): <code>enrole.ldapserver.root=dc=com</code></td>
</tr>
<tr>
<td><code>enrole.ldapserver.home</code></td>
<td>Specifies the location of the system configuration information in the directory server.</td>
</tr>
<tr>
<td></td>
<td>Example (default): <code>enrole.ldapserver.home=ou=itim</code></td>
</tr>
<tr>
<td><code>enrole.ldapserver.agelimit</code></td>
<td>Do not change this property key and value unless you are a qualified administrator. Use the <code>runConfig</code> utility to modify this value.</td>
</tr>
<tr>
<td></td>
<td>Specifies the number of days that an object remains in the recycle bin before it can be deleted when the cleanup script is started. The recycle bin age limit protects objects in the recycle bin from cleanup scripts for the specified length of time.</td>
</tr>
<tr>
<td></td>
<td>Cleanup scripts can remove only those objects that are older than the age limit setting. If the age limit setting is 62 days (default), only objects in the recycle bin for more than 62 days can be deleted by starting the cleanup script.</td>
</tr>
<tr>
<td></td>
<td>Example (default): <code>enrole.ldapserver.agelimit=62</code></td>
</tr>
<tr>
<td><code>enrole.ldapserver.ditlayout</code></td>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td></td>
<td>Specifies the Java class that defines the structure of the data that is stored in the directory server.</td>
</tr>
<tr>
<td></td>
<td>Example (default, flat structure): <code>enrole.ldapserver.ditlayout=com.ibm.itim.dataservices.dit.itim.FlatHashedLayout</code></td>
</tr>
<tr>
<td><code>enrole.ldap.provider</code></td>
<td>Example (default): <code>enrole.ldap.provider=IBM</code></td>
</tr>
</tbody>
</table>
Search and LDAP control properties are used to configure search strategy and LDAP control.

For more information about setting these parameters for your environment, see the tuning guide that is provided for IBM Security Identity Manager.

Table 37 defines the properties used to configure search strategy and LDAP control.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.search.sss.enable</td>
<td>Specifies whether Server Side Sorting is used for searches of the directory server. Enabling server-side sorting with this property can have a large negative impact when you view large organizational units. It is suggested that you disable this option in most environments.</td>
<td>enrole.search.sss.enable=false</td>
</tr>
<tr>
<td>enrole.search.vlv.enable</td>
<td>Specifies whether Virtual List View (VLV) is used for all return data from the directory server. This property can be enabled only when supported by the directory server. This option reduces the memory load on the application server but places a significant load on the LDAP server.</td>
<td>enrole.search.vlv.enable=false</td>
</tr>
<tr>
<td>enrole.search.paging.enable</td>
<td>Specifies whether Paged Sorting is used for searches of the directory server. This option reduces the memory load on the application server. Enabling it is not suggested because the directory server might place a limit on the number of outstanding paged searches.</td>
<td>enrole.search.paging.enable=false</td>
</tr>
<tr>
<td>enrole.search.pagingpagesize</td>
<td>Specifies the page size used for paged LDAP searches when enrole.search.paging.enable=true.</td>
<td>enrole.search.pagingpagesize=128</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td>Example (default)</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>enrole.search.cache.enable</td>
<td>Specifies the use of cached searching to speed up LDAP searches.</td>
<td>enrole.search.cache.enable=true</td>
</tr>
<tr>
<td>enrole.search.cache.secondary.enable</td>
<td>Specifies the use of secondary cached searching to speed up LDAP searches.</td>
<td>enrole.search.cache.secondary.enable=true</td>
</tr>
<tr>
<td>enrole.search.cache.secondary.filter.1</td>
<td>Use a filter fragment for people to prevent LDAP search filters from getting cached. Filtered out LDAP search filters are cached in the secondary cache, if enabled.</td>
<td>enrole.search.cache.secondary.filter.1=ou=people</td>
</tr>
<tr>
<td>enrole.search.cache.secondary.filter.2</td>
<td>Use a filter fragment for accounts to prevent LDAP search filters from getting cached. Filtered out LDAP search filters are cached in the secondary cache, if enabled.</td>
<td>enrole.search.cache.secondary.filter.2=ou=accounts</td>
</tr>
<tr>
<td>enrole.search.cache.secondary.filter.3</td>
<td>Use a filter fragment for the systemuser to prevent LDAP search filters from getting cached. Filtered out LDAP search filters are cached in the secondary cache, if enabled.</td>
<td>enrole.search.cache.secondary.filter.3=ou=systemuser</td>
</tr>
<tr>
<td>enrole.search.cache.secondary.filter.4</td>
<td>Use a filter fragment for orphan accounts to prevent LDAP search filters from getting cached. Filtered out LDAP search filters are cached in the secondary cache, if enabled.</td>
<td>enrole.search.cache.secondary.filter.4=ou=orphans</td>
</tr>
</tbody>
</table>
### Table 37. Search and LDAP control properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.search.clientside.filtering.enable</td>
<td>Do not modify this property key and value. Specifies the use of client-side filtering as a performance alternative on complex LDAP searches. Example (default): <code>enrole.search.clientside.filtering.enable=true</code></td>
</tr>
</tbody>
</table>
| enrole.search.strategy | Do not modify this property key and value. Specifies the Java class that defines the search strategy to process the return data from the directory server. Strategy values include:  
  - `com.ibm.itim.apps.ejb.search.EnumeratedSearch` (process data on demand)  
    Avoids the use of collections, if possible. Maintains a cache of the number of search links multiplied by the page size. The underlying connection is closed when the page cache is filled. Access control items are applied as results are retrieved.  
  - `com.ibm.itim.apps.ejb.search.CollectedSearch` (process all data)  
    This is the previous search mechanism, which converts the search results into a collection and sort it. Applying access control items on the collection as pages are retrieved. The underlying LDAP connection is freed as soon as the results are transformed into a collection. Example (default): `enrole.search.strategy=com.ibm.itim.apps.ejb.search.EnumeratedSearch` |
| enrole.recyclebin.enable | Disable use of the recycle bin for a majority of objects to improve search times. Example (default for new installations): `enrole.recyclebin.enable=false` |

### Person profile properties

Person profile properties identify a person profile.

Table 38 defines the property used to identify a person profile. This property selects the profile by default when you create people or do advanced person searches in the administrative console.

### Table 38. Person profile property

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.personProfile</td>
<td></td>
</tr>
</tbody>
</table>
Table 39. Profile and schema cache properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>enrole.profile.timeout</strong></td>
<td>This property key and value affects performance tuning for IBM Security Identity Manager. Do not change it unless you are a qualified administrator. Specifies the timeout value in minutes for information in the profile section of the cache. Information exceeding this timeout value is removed from the cache. Example (default): enrole.profile.timeout=10</td>
</tr>
<tr>
<td><strong>enrole.schema.timeout</strong></td>
<td>This property key and value affects performance tuning for IBM Security Identity Manager. Do not change it unless you are a qualified administrator. Specifies the timeout value in minutes for information in the schema section of the cache. Information exceeding this timeout value is removed from the cache. Example (default): enrole.schema.timeout=10</td>
</tr>
<tr>
<td><strong>password.attributes</strong></td>
<td>Specifies which attribute is encrypted by the dataservices component. Example (default, on a single line): password.attributes=ersynchpassword erServicePassword erServicePwd1 erServicePwd2 erServicePwd3 erServicePwd4 erADDomainPassword erPersonPassword erNotesPasswdAddCert eritamcred erep6umds</td>
</tr>
<tr>
<td><strong>enrole.reminder.timeout</strong></td>
<td>Do not change this property key and value unless you are a qualified administrator. Specifies the cache interval (in minutes) for a workflow reminder. Example: enrole.reminder.timeout=10</td>
</tr>
</tbody>
</table>
Table 39. Profile and schema cache properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>signedObjectsCacheTimeout</td>
<td>Specifies the cache interval (in hours) for a signed objects.</td>
</tr>
<tr>
<td>Example: signedObjectsCacheTimeout=8</td>
<td>Do not change this property key and value unless you are a qualified administrator.</td>
</tr>
</tbody>
</table>

**Messaging properties**

Messaging properties configure the internal communication between components of the Java Message Service (JMS) used by IBM Security Identity Manager.

Table 40 defines the properties used to configure the internal communication between components of the Java Message Service (JMS) used by IBM Security Identity Manager.

The adjustment of these property values is important to accurate performance tuning and scalability of the IBM Security Identity Manager product. Do not change property values in this section unless you are a qualified administrator.

**Table 40. Messaging properties**

**Message timeout configuration**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.messaging.ttl</td>
<td>Specifies the lifetime in minutes of a message in the queue. A value of zero specifies an unlimited lifetime.</td>
</tr>
<tr>
<td>Example (default):</td>
<td>enrole.messaging.ttl=0</td>
</tr>
</tbody>
</table>

**Messaging queue configuration**

enrole.messaging.managers= \
  enrole.messaging.adhocSyncQueue \ 
  enrole.messaging.workflowQueue \ 
  enrole.messaging.sharedWorkflowQueue \ 
  enrole.messaging.partitioningServiceQueue \ 
  enrole.messaging.remoteServicesQueue \ 
  enrole.messaging.remotePendingQueue \ 
  enrole.messaging.mailServicesQueue \ 
  enrole.messaging.policyAnalysisQueue \ 
  enrole.messaging.policySimulationQueue \ 
  enrole.messaging.importExportQueue

Do not modify these property keys and values.

Specifies the key names of supported IBM Security Identity Manager queues.
Table 40. Messaging properties  (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.messaging.adhocSyncQueue</td>
<td>adhocSyncQueue</td>
</tr>
<tr>
<td>enrole.messaging.workflowQueue</td>
<td>workflowQueue</td>
</tr>
<tr>
<td>enrole.messaging.sharedWorkflowQueue</td>
<td>sharedWorkflowQueue</td>
</tr>
<tr>
<td>enrole.messaging.partitioningServiceQueue</td>
<td>partitioningServiceQueue</td>
</tr>
<tr>
<td>enrole.messaging.remoteServicesQueue</td>
<td>remoteServicesQueue</td>
</tr>
<tr>
<td>enrole.messaging.remotePendingQueue</td>
<td>remotePendingQueue</td>
</tr>
<tr>
<td>enrole.messaging.mailServicesQueue</td>
<td>mailServicesQueue</td>
</tr>
<tr>
<td>enrole.messaging.policyAnalysisQueue</td>
<td>policyAnalysisQueue</td>
</tr>
<tr>
<td>enrole.messaging.policySimulationQueue</td>
<td>policySimulationQueue</td>
</tr>
<tr>
<td>enrole.messaging.importExportQueue</td>
<td>importExportQueue</td>
</tr>
</tbody>
</table>

Do not modify these property keys and values.

Specifies the actual queue name as referenced by the application server.

Queue attribute configuration

- **SHARED**
  
  A Boolean value that indicates whether the queue is shared across a clustered deployment. In a cluster, a shared queue can be read and written to by all cluster members.
  
  Do not modify this property.

  Example (on a single line):

  ```
  enrole.messaging.sharedWorkflowQueue.attributes=SHARED=true
  enrole.messaging.policyAnalysisQueue.attributes=SHARED=true
  enrole.messaging.policySimulationQueue.attributes=SHARED=true
  ```

Message processing errors detected by the messaging system cause individual messages to be redelivered and additional attempts to handle the message. Following the first indication of process failure, a retry is scheduled immediately. If the first attempt fails, another is scheduled with a delay that matches the value of the `FIRST_RETRY_DELAY` property. If the second attempt fails, another is scheduled with a delay that matches the value of the `RETRY_DELAY` property. Subsequent retries are attempted with the value of the `RETRY_DELAY` property until the `MAX_RETRY_TIME` threshold is reached.

Set the following properties to manage how the system handles the retry attempts.

- **FIRST_RETRY_DELAY**
  
  The amount of time in milliseconds to delay before retrying after the initial immediate retry. Default value is 900000 (15 minutes).

- **RETRY_DELAY**
  
  The amount of time [in milliseconds] to delay before retrying after the immediate and first attempts fail. Default value is 3600000 (60 minutes).

- **MAX_RETRY_TIME**
  
  The maximum amount of time allowed for attempts, beginning with the first failure. Default value is 86400000 (24 hours)

Example (on a single line):

```
enrole.messaging.workflowQueue.attributes=SHARED=false
FIRST_RETRY_DELAY=300000 RETRY_DELAY=900000 MAX_RETRY_TIME=3600000
```

Scheduling properties

The scheduling properties are used to configure the internal scheduler that runs calendar-based and scheduled events.
Table 41 defines the properties used to configure the internal scheduler responsible for running calendar-based scheduled events. Events and their schedules are stored in a database table.

Table 41. Scheduling properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.scheduling.heartbeat</td>
<td>This property key and value affects performance tuning for IBM Security Identity Manager. Do not change it unless you are a qualified administrator. Specifies the interval [in seconds] that the event monitor checks the database table for scheduled events. Example (default): enrole.scheduling.heartbeat=30</td>
<td></td>
</tr>
<tr>
<td>enrole.scheduling.timeout</td>
<td>This property key and value affects performance tuning for IBM Security Identity Manager. Do not change it unless you are a qualified administrator. Specifies the timeout value [in minutes] for the event processor. Example (default): enrole.scheduling.timeout=10</td>
<td></td>
</tr>
<tr>
<td>enrole.scheduling.fetchsize</td>
<td>This property key and value affects performance tuning for IBM Security Identity Manager. Do not change it unless you are a qualified administrator. Specifies the number of messages to retrieve at a time when in batch mode. Example (default): enrole.scheduling.fetchsize=50</td>
<td></td>
</tr>
</tbody>
</table>

Password transaction monitor properties

Password transaction monitor properties checks responses to password transactions. It expires those transactions when the user fails to respond in the specified interval.

When a password for a user is changed or automatically generated, an email notification is sent to a user. The email contains either the actual password or a link that the user can follow to obtain the new password. This activity is called a password transaction. The user must respond to the email and incorporate the new password within a specified amount of time. If the user fails to respond within the allowed time period, the password transaction expires.

The password transaction monitor is responsible for checking responses to password transactions. It expires those transactions when the user fails to respond to the email.

Table 42. Password transaction monitor properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.passwordtransactionmonitor.heartbeat</td>
<td></td>
</tr>
</tbody>
</table>
Table 42. Password transaction monitor properties (continued)

<table>
<thead>
<tr>
<th></th>
<th>Specifies how often [in hours] the password transaction monitor checks for expired password transactions.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example (default):</td>
</tr>
<tr>
<td></td>
<td>enrole.passwordtransactionmonitor.heartbeat=1</td>
</tr>
</tbody>
</table>

XML and DTD properties

XML and DTD properties are no longer used.

These properties are no longer used.

Table 43. XML and DTD properties

| enrole.dtd.uri  | Not used.                                                  |

LDAP connection pool properties

LDAP connection pool properties are used to configure cache connection requests to the directory server.

Table 44 defines the properties used to configure the values that affect cache connection requests to the IBM Security Identity Manager directory server.

Table 44. LDAP connection pool properties

<table>
<thead>
<tr>
<th>enrole.connectionpool.incrementcount</th>
<th>This property key and value affect performance tuning for IBM Security Identity Manager. They must be changed only by a qualified administrator. Specifies the number of connections that are created any time the LDAP connection pool is incremented to accommodate an increasing demand.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example (default):</td>
</tr>
<tr>
<td></td>
<td>enrole.connectionpool.incrementcount=3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enrole.connectionpool.authentication</th>
<th>This property key and value affect performance tuning for IBM Security Identity Manager. They must be changed only by a qualified administrator. Specifies a list of space-separated authentication types of connections that can be pooled. Valid types are:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• none - No authentication is required.</td>
</tr>
<tr>
<td></td>
<td>• simple</td>
</tr>
<tr>
<td></td>
<td>• DIGEST-MD5 -</td>
</tr>
<tr>
<td></td>
<td>Example (default):</td>
</tr>
<tr>
<td></td>
<td>enrole.connectionpool.authentication=none simple</td>
</tr>
</tbody>
</table>
Table 44. LDAP connection pool properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| enrole.connectionpool.debug   | This property key and value specify the level of debug output. Valid values are “fine” (trace connection creation and removal) and “all” (all debugging information). Valid values are:  
  • **fine** - Trace connection creation and removal.  
  • **all** - All debugging information.  
  Example (default, commented out):  
  #enrole.connectionpool.debug=fine  |
| enrole.connectionpool.initialpoolsize | This property key and value affect performance tuning for IBM Security Identity Manager. They must be changed only by a qualified administrator. Specifies the initial number of physical LDAP connections to create for the LDAP connection pool. This value must be less than or equal to the value of the maxpoolsize property.  
  Example (default):  
  enrole.connectionpool.initialpoolsize=50  |
| enrole.connectionpool.maxpoolsize | This property key and value affect performance tuning for IBM Security Identity Manager. They must be changed only by a qualified administrator. Specifies the maximum number of physical LDAP connections that can be created.  
  Example (default):  
  enrole.connectionpool.maxpoolsize=100  |
| enrole.connectionpool.prefsize | This property key and value affect performance tuning for IBM Security Identity Manager. They must be changed only by a qualified administrator. Specifies the preferred number of physical LDAP connections that must be maintained concurrently. This number includes both in-use and idle connections. A size of zero or no value means that there is no preferred size. In that case, a request for a pooled connection results in a newly created connection if no idle ones are available.  
  Example (no value):  
  enrole.connectionpool.prefsize=  |
| enrole.connectionpool.protocol | |
Password encryption properties

Password encryption properties are used to configure password encryption.

Table 45 defines the properties used to configure password encryption.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.encryption.algorithm</code></td>
<td>Specifies the cipher suite to use for encryption. For example, AES or PBEWithMD5AndDES.</td>
</tr>
<tr>
<td></td>
<td>Example (default):</td>
</tr>
<tr>
<td></td>
<td><code>enrole.encryption.algorithm=AES</code></td>
</tr>
<tr>
<td><code>enrole.encryption.password</code></td>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td></td>
<td>Specifies the password encryption password.</td>
</tr>
<tr>
<td></td>
<td>Example (default):</td>
</tr>
<tr>
<td></td>
<td><code>enrole.encryption.password=AES</code></td>
</tr>
</tbody>
</table>
Table 45. Encryption properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.encryption.password</code></td>
<td>Do not modify this property key and value. This value is specified during IBM Security Identity Manager installation. The value of the <code>enrole.encryption.password</code> property is moved into the <code>encryptionKey</code> property file. The value is encoded by default and is stored in the <code>encryptionKey</code> property file. For Password-Based Encryption (PBE) encryption algorithms (used for upgraded IBM Tivoli Identity Manager Version 4.6 installations), specifies the password used as an input parameter for Password-Based Encryption (PBE). PBE is a method of encrypting and decrypting data with a secret key based on a user-supplied password. For example, encrypted data includes shared secrets, service passwords, and some protected account attributes. Specifies the keystore password when AES is the encryption algorithm. For non-PBE based encryption algorithms (used for new IBM Tivoli Identity Manager Version 5.0 installations), the password is used to encrypt the keystore that stores the private key. For more information about this property, see the <code>enrole.encryption.keystore</code> property. This value is specified during IBM Security Identity Manager installation.</td>
</tr>
<tr>
<td><code>enrole.encryption.passwordDigest</code></td>
<td>Do not modify this property key and value. Specifies the type of password digest used for an IBM Security Identity Manager password. Upgrading Tivoli Identity Manager from Version 4.6 continues to use the original hash algorithm until users change their passwords. This original algorithm is defined by the property <code>enrole.pre50.encryption.passwordDigest</code>. Valid values are:</td>
</tr>
<tr>
<td><code>enrole.pre50.encryption.passwordDigest</code></td>
<td>Do not modify this property key and value. Upgrading IBM Tivoli Identity Manager from Version 4.6 adds this property dynamically to this properties file. Specifies the type of password digest used for IBM Security Identity Manager password data from IBM Tivoli Identity Manager versions before 5.0. The lack of a <code>:</code> in an encrypted IBM Security Identity Manager password value is used to identify such migrated data. <strong>Note:</strong> All new passwords, including changed migrated passwords, are stored with the <code>enrole.encryption.passwordDigest</code> algorithm. Example (default for migrated installations, not present for new installations): <code>enrole.pre50.encryption.passwordDigest=MD5</code></td>
</tr>
</tbody>
</table>

---

Example (default): `enrole.encryption.password=foo`
**Table 45. Encryption properties (continued)**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.encryption.keystore</code></td>
<td>Do not modify this property key and value. Specify the keystore file name used to contain the randomly generated secret key for non-PBE based encryption algorithms, such as AES. This keystore file is protected with the <code>enrole.encryption.password</code> value. This file is in the <code>ISIM_HOME\data\keystore</code> directory. Example (default): <code>enrole.encryption.keystore=itimKeystore.jceks</code></td>
</tr>
</tbody>
</table>

**Challenge response encoding properties**

Challenge response encoding properties determine whether a response is encoded as case sensitive or insensitive.

**Table 46. Challenge response encoding properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| `enrole.challengeresponse.responseConvertCase` | Do not change this property key and value unless you are a qualified administrator. Specifies how CR responses are encoded before they are stored in the directory. Valid values are:  
  • `lower` – Encode the CR as lowercase.  
  • `upper` – Encode the CR as uppercase.  
  • `none` – Do not encode the CR. Retain the case-sensitive response as is.  
Example (default): `enrole.challengeresponse.responseConvertCase=lower` |

**System listening port properties**

System listening port properties are used to configure the listening port settings for the IBM Security Identity Manager Server.

**Table 47. System configuration properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.system.listenPort</code></td>
<td>Do not modify this property key and value. Specifies the TCP (non-secure communication) listening port value. This value is set during IBM Security Identity Manager installation. Example (default): <code>enrole.system.listenPort=80</code></td>
</tr>
</tbody>
</table>
Table 47. System configuration properties (continued)

| enrole.system.SSLlistenPort | Do not modify this property key and value. Specifies the Secure Sockets Layer (SSL) listening port value. This value is set during IBM Security Identity Manager installation. Example (default): enrole.system.SSLlistenPort=443 |

Mail properties

Mail properties are used to configure internal mail notification. Table 48 defines the properties used to configure internal mail notification.

Table 48. Mail services properties

| enrole.mail.notify | Specifies whether the sending of workflow internal email is synchronized or not. Values include: • SYNC - Synchronized. • ASYNC - Asynchronized. Example (default): enrole.mail.notify=ASYNC |

Workflow properties

Workflow properties are used to configure the core IBM Security Identity Manager workflow engine. Table 49 on page 251 defines the properties used to configure the core IBM Security Identity Manager workflow engine.

Note: If you begin your upgrade to Version 5.0 from Tivoli Identity Manager Version 4.5.x, and then to Version 4.6, the workflow notification properties are not modified during the upgrade. To have notification template customization available in IBM Tivoli Identity Manager Version 4.6 following an upgrade, you must modify the values of these properties. You must modify them to the new Template notification factories (prefixed with Template).

For example, the enrole.workflow.notification.activitytimeout property for Tivoli Identity Manager Version 4.5.x is shown in the following example (on a single line).

enrole.workflow.notification.activitytimeout=
com.ibm.itim.workflow.notification.ActivityTimeoutNotification

If you upgrade Tivoli Identity Manager Version 4.6 to Version 5.0, the change occurs automatically. It assumes either of the following conditions:
• Version 4.6 was the starting point for upgrade
You made the manual change to the `enrole.workflow.notification.activitytimeout` property before you upgrade from Version 4.5.x

### Table 49. Workflow configuration properties

<table>
<thead>
<tr>
<th>Workflow configuration</th>
<th>Description</th>
</tr>
</thead>
</table>
| **enrole.workflow.lrucache.size** | Specifies the size of the cache used to temporarily use and access workflow objects. Do not change it unless directed by IBM support. Making this value too large can result in out of memory conditions of IBM Security Identity Manager Server.  
Example (default, commented out): 
```
## enrole.workflow.lrucache.size=number_of_entries
```
where the default value of `number_of_entries` is 2000. |
| **enrole.workflow.notifyoption** | Do not change this property key and value unless you are a qualified administrator.  
Specifies the behavior of workflow email notifications. Values are:  
- 0 (NOTIFY_NONE) – IBM Security Identity Manager does not send email notifications when the workflow process completes.  
- 1 (NOTIFY_REQUESTER) – A process completion notification is sent to the requester when the workflow process completes. Account email notifications are then sent to the requestee for the following account requests:  
  - New Account  
  - New Password  
  - Change Account  
  - Deprovision Account  
  - Suspend Account  
  - Restore Account  
For example, when the workflow process completes for a new account request, a process completion notification is sent to the requester. A new account notification is then sent to the requestee.  
Example (default): 
enrole.workflow.notifyoption=1 |
| **enrole.workflow.notifypassword** | Do not change this property key and value unless you are a qualified administrator.  
Specifies the type of email notification in a password transaction (caused when a user password is changed or automatically generated). Values are:  
- **true** – email notification of a password change can be sent to a user. The actual notification mechanism and whether to include the actual password in the email is dictated by the configuration of the `enrole.workflow.notification.newpassword` property value.  
- **false** – email notification of a password change is sent to a user. The email contains a URL where the user can obtain the password. The URL prompts the user for the shared secret.  
Example (default): 
enrole.workflow.notifypassword=true |
### Table 49. Workflow configuration properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.workflow.notifyaccountsonwarning</code></td>
<td>Specifies whether account email notifications are sent when the account operation results in a warning. Values are:</td>
<td><code>enrole.workflow.notifyaccountsonwarning=false</code></td>
</tr>
<tr>
<td></td>
<td>• <strong>true</strong> – Sends account email notifications.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>false</strong> – Does not send account email notifications.</td>
<td></td>
</tr>
<tr>
<td><code>enrole.workflow.maxretry</code></td>
<td>Do not change this property key and value unless you are a qualified administrator.</td>
<td><code>enrole.workflow.maxretry=2</code></td>
</tr>
<tr>
<td></td>
<td>Specifies the number of times an attempt is made to start a workflow that initially failed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>See also <code>enrole.workflow.retrydelay</code>.</td>
<td></td>
</tr>
<tr>
<td><code>enrole.workflow.retrydelay</code></td>
<td>Do not change this property key and value unless you are a qualified administrator.</td>
<td><code>enrole.workflow.retrydelay=60000</code></td>
</tr>
<tr>
<td></td>
<td>Specifies the time delay [in milliseconds] between successive attempts to start a workflow application that initially failed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>See also <code>enrole.workflow.maxretry</code>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example (default):</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>enrole.workflow.retrydelay=60000</code></td>
<td></td>
</tr>
<tr>
<td><code>enrole.workflow.skipapprovalforrequester</code></td>
<td>For a workflow activity that requires approval, this property specifies whether to skip the approval for other approvers if the requester is also an approver. Values are:</td>
<td><code>enrole.workflow.skipapprovalforrequester=false</code></td>
</tr>
<tr>
<td></td>
<td>• <strong>true</strong> – Skips approval for other approvers if the requester is also an approver.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>false</strong> – Forces an approval check from other required approvers of the activity, except the requester (if the requester is also an approver). If the requester is a single approver as a result of participant resolution, then the approval is skipped even when value is set to false.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example (default):</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>enrole.workflow.skipapprovalforrequester=false</code></td>
<td></td>
</tr>
<tr>
<td><code>enrole.workflow.disablerequesteeapproval</code></td>
<td>Do not change this property key and value unless you are a qualified administrator.</td>
<td></td>
</tr>
</tbody>
</table>
**Table 49. Workflow configuration properties (continued)**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| **enrole.workflow.disablerequesteeapproval** | Do not change this property key and value unless you are a qualified administrator. For a workflow activity that requires approval, this property specifies whether to disable the requestee approval if the requestee is also an approver. Values are:  
  - **true** – Disables the requestee approval if the requestee is also an approver.  
  - **false** – Sends an approval check to the requestee and other resolved participants if the requestee is also an approver.  
  The default value is **false**.  
  Example (default):  
  `enrole.workflow.disablerequesteeapproval=false`  
  For more information, see Planning > Workflow planning > Workflow participants > Disable requestee or requester approval on the IBM Security Identity Manager documentation. |
| **enrole.workflow.disablerequesterapproval** | Do not change this property key and value unless you are a qualified administrator. IBM Security Identity Manager considers this property value only when the `enrole.workflow.skipapprovalforrequester` property value is set to **false**. For a workflow activity that requires approval, this property specifies whether to disable the requester approval if the requester is an approver. Values are:  
  - **true** – A value set to **false** for the `enrole.workflow.skipapprovalforrequester` property disables automatic approval if the requester is a lone approver.  
  - **false** – Works according to the value that you set for the `enrole.workflow.skipapprovalforrequester` property.  
  Example (default):  
  `enrole.workflow.disablerequesterapproval=false`  
  For more information, see Planning > Workflow planning > Workflow participants > Disable requestee or requester approval on the IBM Security Identity Manager documentation. |
| **enrole.workflow.skipfornoncompliantaccount** | Do not change this property key and value unless you are a qualified administrator. Specifies whether to engage the entitlement workflow that is associated with the account. Specifies when a system account modification is triggered as a result of a policy enforcement action. Values are:  
  - **true** – Skips this action.  
  - **false** – Does not skip this action.  
  Example (default):  
  `enrole.workflow.skipfornoncompliantaccount=true` |
<p>| <strong>enrole.workflow.distribution</strong> | Chapter 16. System property configuration in enRole.properties | 253 |</p>
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.workflow.async_completion_enabled</td>
<td>Specifies whether the system uses asynchronous completion checking for some system workflows. Only applicable when enrole.workflow.async_completion_enabled=true.</td>
<td></td>
</tr>
<tr>
<td>enrole.workflow.async_completion_interval_sec</td>
<td>Specifies the interval in seconds that the system checks to see whether certain system workflows are complete.</td>
<td></td>
</tr>
<tr>
<td>enrole.workflow.notification.activitytimeout</td>
<td>Specifies the default Java class that generates the workflow activity timeout notification.</td>
<td></td>
</tr>
<tr>
<td>enrole.workflow.notification.processtimeout</td>
<td>Specifies the default Java class that generates the workflow process timeout notification.</td>
<td></td>
</tr>
</tbody>
</table>
| enrole.workflow.notification.processcomplete  | Do not change this property key and value unless you are a qualified administrator. Specifies whether workflow requests use the IBM Security Identity Manager shared queues, which allow for workload distribution. Values are:  
  • true – Workflow requests are eligible for distribution.  
  • false – Workflow requests are not eligible for distribution. |

Example (default): enrole.workflow.distribution=true
Table 49. Workflow configuration properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example (default, entered as a single line)</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.workflow.notification.processcomplete</td>
<td>Specifies the default Java class that generates the notification for when a workflow process is completed.</td>
<td>enrole.workflow.notification.processcomplete=com.ibm.itim.workflow.notification.TemplateProcessCompleteNotification</td>
</tr>
<tr>
<td>enrole.workflow.notification.pendingwork</td>
<td>Specifies the default Java class that generates the notification for when a workflow process is completed for manual activities (Approvals and Requests for Information).</td>
<td>enrole.workflow.notification.pendingwork=com.ibm.itim.workflow.notification.TemplatePendingWorkNotification</td>
</tr>
<tr>
<td>enrole.workflow.notification.newaccount</td>
<td>Specifies the default Java class that generates the notification for when a workflow process is completed for a new account.</td>
<td>enrole.workflow.notification.newaccount=com.ibm.itim.workflow.notification.TemplateNewAccountNotification</td>
</tr>
<tr>
<td>enrole.workflow.notification.newpassword</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Chapter 16. System property configuration in enRole.properties 255
Do not change this property key and value unless you are a qualified administrator.

Specifies the default Java class that generates a notification when a user changes a password. This property is used only when the value for the property is true.

```
enrole.workflow.notifypassword=true
```

This property responds to the following three-password change scenarios:

- When a user changes the password for the account
- When the administrator forces a password change on the account
- When a user is successfully identified through the password challenge/response feature, and challenge/response is configured.

Valid classes include:

- **NewPasswordNotification**
  
  Email notification that includes the password (in ASCII text) is sent to a user (default).

- **EmptyNotificationFactory**
  
  Suppresses email notification. The preferred method for suppressing any notification is through the Workflow Notification GUI.

- **PasswordChangeNotificationFactory**
  
  Email notification that does not include the password is sent to a user. Message body says: "Process completed".

The `EmptyNotificationFactory` and `PasswordChangeNotificationFactory` classes are in the `examples.jar` package in the `examples` directory.

Example (default, entered as a single line):

```
enrole.workflow.notification.newpassword=com.ibm.itim.workflow.notification.TemplateNewPasswordNotification
```

### `enrole.workflow.notification.deprovision`

Do not change this property key and value unless you are a qualified administrator.

Specifies the default Java class that generates deprovisioning notification.

Example (default, entered as a single line):

```
enrole.workflow.notification.deprovision=com.ibm.itim.workflow.notification.TemplateDeprovisionNotification
```

### `enrole.workflow.notification.workorder`

Do not change this property key and value unless you are a qualified administrator.

Specifies the default Java class that generates work order notifications.

Example (default, entered as a single line):

```
enrole.workflow.notification.workorder=com.ibm.itim.workflow.notification.TemplateWorkOrderNotification
```

### `enrole.workflow.notification.changeaccount`

Do not change this property key and value unless you are a qualified administrator.
Table 49. Workflow configuration properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| enrole.workflow.notification.changeaccount | Specifies the default Java class that generates account change notifications. Example (default, as a single line):
| enrole.workflow.notification.changeaccount=com.ibm.itim.workflow.notification.TemplateChangeAccountNotification |

enrole.workflow.notification.restoreaccount

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| enrole.workflow.notification.restoreaccount | Specifies the default Java class that generates account restoration notifications. Example (as a single line):
| enrole.workflow.notification.restoreaccount=com.ibm.itim.workflow.notification.TemplateRestoreAccountNotification |

enrole.workflow.notification.suspendaccount

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| enrole.workflow.notification.suspendaccount | Specifies the default Java class that generates account suspension notifications. Example (as a single line):
| enrole.workflow.notification.suspendaccount=com.ibm.itim.workflow.notification.TemplateSuspendAccountNotification |

Reconciliation properties

Reconciliation properties are used to configure the reconciliation process where data retrieved from agents is synchronized in the IBM Security Identity Manager database.

Table 50 defines the properties used to configure the values that affect the reconciliation process where data retrieved from agents is synchronized in the IBM Security Identity Manager database.

Table 50. Reconciliation properties

<table>
<thead>
<tr>
<th>Reconciliation configuration</th>
<th>Description</th>
</tr>
</thead>
</table>
| enrole.reconciliation.accountcachesize | Specifies the maximum size of the cache for existing accounts cache that is used for the reconciliation process. Setting a value larger than the default might cause processing of reconciliations to fail. Example (default):
| enrole.reconciliation.accountcachesize=2000 |

enrole.reconciliation.threadcount
Table 50. Reconciliation properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.reconciliation.threadcount</code></td>
<td>Specifies the number of threads that are used to handle reconciled entries. This number of threads is created for each reconciliation process. Example (default): enrole.reconciliation.threadcount=8</td>
</tr>
<tr>
<td><code>enrole.reconciliation.failurethreshold</code></td>
<td>Specifies the maximum number of local accounts to delete at the end of reconciliation. If the value is exceeded, then no local account or supporting data entries are deleted. Example (default, commented out): #enrole.reconciliation.failurethreshold=100%</td>
</tr>
<tr>
<td><code>enrole.reconciliation.logTimeInterval</code></td>
<td>Specifies the time interval in seconds for reconciliation progress trace log messages. A value of zero disables this time interval. Example (default, commented out): #enrole.reconciliation.logTimeInterval=600</td>
</tr>
<tr>
<td><code>enrole.reconciliation.logEveryNResults</code></td>
<td>Specifies the count for reconciliation progress trace log messages. A value of zero disables this count. Example (default, commented out): #enrole.reconciliation.logEveryNResults=5000</td>
</tr>
</tbody>
</table>

Unsolicited notification events

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>person.EventProcessorFactory</code></td>
<td>Do not modify this property key and value.</td>
</tr>
</tbody>
</table>
### Table 50. Reconciliation properties (continued)

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Example (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Do not modify this property key and value.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>person.EventProcessorFactory</td>
<td>Specifies the built-in Java class for the person event processor factory.</td>
<td>person.EventProcessorFactory=com.ibm.itim.remoteservices.ejb.reconciliation.PersonEventProcessorFactory</td>
</tr>
<tr>
<td>person.ReconEntryHandlerFactory</td>
<td>Specifies the built-in Java class for the person entry handler factory.</td>
<td>person.ReconEntryHandlerFactory=com.ibm.itim.remoteservices.ejb.mediation.PersonEntryHandlerFactory</td>
</tr>
<tr>
<td><strong>Deferring requests for failed remote resources</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>com.ibm.itim.remoteservices.ResourceProperties.DEFER_FAILED_RESOURCE</td>
<td>Do not modify this property key and value. Specifies whether to defer requests to failed resources and wait for resource to restart before it sends them. Valid values are:</td>
<td>com.ibm.itim.remoteservices.ResourceProperties.DEFER_FAILED_RESOURCE=true</td>
</tr>
<tr>
<td>remoteservices.remotepending.interval</td>
<td>Specifies the interval in seconds (120 minimum to 3600 maximum) to check whether failed resources restart.</td>
<td>remoteservices.remotepending.interval=600</td>
</tr>
</tbody>
</table>
Table 50. Reconciliation properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not modify this property key and value. Specifies the maximum time in seconds that a request to a resource can be outstanding. It includes time in pending state for asynchronous requests, or deferred requests due to a service failure or request backlog. Valid values are:</td>
<td></td>
</tr>
<tr>
<td>-1 – Unlimited</td>
<td></td>
</tr>
<tr>
<td>(60 + (\text{value of remoteservices.remotepending.interval})) – Minimum time interval for outstanding requests.</td>
<td></td>
</tr>
<tr>
<td>Example (default):</td>
<td></td>
</tr>
<tr>
<td>com.ibm.itim.remoteservices.ResourceProperties.MAX_REQUEST_TIME=-1</td>
<td></td>
</tr>
</tbody>
</table>

remoteservices.remotepending.restart.retry

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not modify this property key and value. Specifies the time interval in minutes that pending requests generated from the restart of a failed service are given to complete. When the time interval ends, the server retries the requests.</td>
<td></td>
</tr>
<tr>
<td>Example (default):</td>
<td></td>
</tr>
<tr>
<td>remoteservices.remotepending.restart.retry=1440</td>
<td></td>
</tr>
</tbody>
</table>

com.ibm.itim.remoteservices.DSML2ServiceProvider.modifyAsREPLACE

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not change this property key and value unless you are a qualified administrator. For remote services, specifies the DSMLv2 (deprecated) provider mode of sending a modify request for attributes.</td>
<td></td>
</tr>
<tr>
<td>Values include:</td>
<td></td>
</tr>
<tr>
<td>true – Use the REPLACE operation.</td>
<td></td>
</tr>
<tr>
<td>false – Use the ADD and DELETE operations.</td>
<td></td>
</tr>
<tr>
<td>Example (default):</td>
<td></td>
</tr>
<tr>
<td>com.ibm.itim.remoteservices.DSML2ServiceProvider.modifyAsREPLACE=true</td>
<td></td>
</tr>
</tbody>
</table>

Shared secret properties

Shared secret properties are used to configure the level of protection of the shared secret code.

Table 51 defines the properties used to configure the level of protection of the shared secret code.

The shared secret is used by an account owner to retrieve a new or changed password for an account when the system is configured to not email passwords in the clear (that is, the value of enrole.workflow.notifypassword=false). This property determines whether the stored shared secret is hashed for additional protection.

Table 51. Shared secret hashing properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.sharedsecret.hashed</td>
<td></td>
</tr>
</tbody>
</table>
Table 51. Shared secret hashing properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| `enrole.sharedsecret.hashed` | Specifies whether the shared secret code is hashed (secure) or not hashed (not secure). Values include:
| `true` | Store the shared secret as hashed. |
| `false` | Store the shared secret as not hashed. |
| Example (default): | `enrole.sharedsecret.hashed=false` |

Lifecycle rule properties

Lifecycle rule properties define values such as the partition size used for lifecycle rules.

Table 52 defines the properties used to configure lifecycle rules.

Table 52. Lifecycle rule properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.lifecyclerule.partition.size</code></td>
<td>Specifies the size of the data partitions for processing lifecycle rules. This parameter determines how much data is processed in a single step. Example (default): <code>enrole.lifecyclerule.partition.size=100</code></td>
</tr>
</tbody>
</table>

Product name properties

Product name properties identify this product.

Table 53 defines the property used to identify the product.

Table 53. Product property

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.product.name</code></td>
<td>Do not change this name. This property key identifies the product name as IBM Security Identity Manager. Example (default): <code>enrole.product.name=ITIM Enterprise</code></td>
</tr>
</tbody>
</table>

Application client request properties

Application client request properties define the properties used to configure the lifetime, or timeout, value for the authentication token used to allow third-party communication with IBM Security Identity Manager Server.

Table 54 on page 262 defines the properties used to configure the lifetime, or timeout, value for the authentication token used by the IBM Security Identity Manager Server.
Manager application API to allow third-party applications to communicate with the IBM Security Identity Manager Server.

### Table 54. Application client request properties

<table>
<thead>
<tr>
<th>authTokenTimeout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies timeout value in hours for the authentication token that is used for communication between third-party applications (with the IBM Security Identity Manager application API) and the IBM Security Identity Manager Server.</td>
</tr>
<tr>
<td>A value of -1 indicates that there is no timeout for the authentication token.</td>
</tr>
<tr>
<td>Example (default):</td>
</tr>
<tr>
<td>authTokenTimeout=48</td>
</tr>
</tbody>
</table>

---

### Reverse password synchronization properties

Reverse password synchronization properties are used to configure reverse password synchronization.

Table 55 defines the properties used to configure reverse password synchronization.

#### Table 55. Reverse password synchronization properties

<table>
<thead>
<tr>
<th>reversePasswordSynch.bypassPwdValidationOnOrphanAccount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies whether to bypass the password validation on the orphan account when the request is submitted from the agent. Valid values are:</td>
</tr>
<tr>
<td>• true – Bypass password validation.</td>
</tr>
<tr>
<td>• false – Validate passwords.</td>
</tr>
<tr>
<td>Example (default):</td>
</tr>
<tr>
<td>reversePasswordSynch.bypassPwdValidationOnOrphanAccount=false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enrole.passwordsynch.module.sendMail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies whether to enable or disable email notifications when password synchronization is triggered by the reverse password synchronization agent, not from the IBM Security Identity Manager graphical user interface. Valid values are:</td>
</tr>
<tr>
<td>• true – Enable email notifications.</td>
</tr>
<tr>
<td>• false – Disable email notifications.</td>
</tr>
<tr>
<td>Example (default):</td>
</tr>
<tr>
<td>enrole.passwordsynch.module.sendMail=false</td>
</tr>
</tbody>
</table>

---

### Post office properties

Post office properties are used to configure the post office for email collection.

Table 56 defines the properties for testing post office configuration.

#### Table 56. Post office properties

- enrole.postoffice.test.subject1
- enrole.postoffice.test.textbody1
- enrole.postoffice.test.xhtmlbody1
Table 56. Post office properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.postoffice.test.subject1</td>
<td>Specifies the contents of the emails that are used when you test the post office configuration. It is one of three emails to which the template is applied. Example (default): enrole.postoffice.test.subject1=This is subject 1 enrole.postoffice.test.textbody1=This is the text body 1 enrole.postoffice.test.xhtmlbody1=This is the xhtml body 1</td>
</tr>
<tr>
<td>enrole.postoffice.test.subject2</td>
<td>Specifies the contents of the emails that are used when you test the post office configuration. It is one of three emails to which the template is applied. Example (default): enrole.postoffice.test.subject2=This is subject 2 enrole.postoffice.test.textbody2=This is the text body 2 enrole.postoffice.test.xhtmlbody2=This is the xhtml body 2</td>
</tr>
<tr>
<td>enrole.postoffice.test.subject3</td>
<td>Specifies the contents of the emails that are used when you test the post office configuration. It is one of three emails to which the template is applied. Example (default): enrole.postoffice.test.subject3=This is subject 3 enrole.postoffice.test.textbody3=This is the text body 3 enrole.postoffice.test.xhtmlbody3=This is the xhtml body 3</td>
</tr>
<tr>
<td>enrole.postoffice.test.topic</td>
<td>Specifies the topic of the email that is used when you test the post office configuration. The three test emails, whose content is defined by the preceding properties, all have this topic. The post office function gathers and stores emails by topic and locale. It then aggregates and sends them as one email on a configured interval, such as once a day or once a week. This method prevents flooding the recipient with many individual emails for a type of event. The topic data usually indicates the type of event. It is also made available to the programming environment that is activated when the gathered emails are aggregated into one summarizing email. In this way, the topic under which all of these emails were gathered can be prominently displayed in the aggregate email that is sent. Example (default): enrole.postoffice.test.topic=topic1</td>
</tr>
<tr>
<td>enrole.postoffice.test.locale</td>
<td>Specifies the locale for the language that is used in an email. Example (default): enrole.postoffice.test.locale=en_US</td>
</tr>
</tbody>
</table>
**Database resource bundle properties**

Database resource bundle properties determine the refresh interval for the database resource bundle.

Table 57 defines the properties used to determine the refresh interval for the database resource bundle.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.databaseresourcebundle.refreshInterval</td>
<td>Specifies how many minutes to wait before DatabaseResourceBundle is checked for changes and reloaded.</td>
<td>enrole.databaseresourcebundle.refreshInterval=5</td>
</tr>
</tbody>
</table>

**Database cleanup properties**

Database cleanup properties define the parameters to clean up session information in the database.

Table 58 defines the parameters for the policy analysis scavenger thread to clean up session information in the database.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>provisioning.policy.preview.cleanup.interval</td>
<td>Specifies the interval in minutes that the scavenger thread scans the database.</td>
<td>provisioning.policy.preview.cleanup.interval=30</td>
</tr>
<tr>
<td>provisioning.policy.analysis.idle.timeout</td>
<td>Represents the expired time setting for a policy analysis session. The scavenger thread cleans up the staged data of a policy analysis session if the session ends at an interval that is greater than the timeout value. The timeout value might be 120 minutes.</td>
<td>provisioning.policy.analysis.idle.timeout=120</td>
</tr>
</tbody>
</table>

**Account restore properties**

Account restore properties can suppress the need for a new password when an account is restored.

Table 59 defines the properties for account restore configuration.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>account.restore.password.suppress</td>
<td></td>
</tr>
</tbody>
</table>
Table 59. Account restore properties (continued)

<table>
<thead>
<tr>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disables prompting for a new password when you restore accounts.</td>
</tr>
<tr>
<td>This property must be manually added to the enRole.properties file if you</td>
</tr>
<tr>
<td>implement this feature. Even when this property is set to true, some accounts</td>
</tr>
<tr>
<td>still require a new password during restoration.</td>
</tr>
<tr>
<td>To disable prompting for a new password during restoration, stop the IBM</td>
</tr>
<tr>
<td>Security Identity Manager Server, add (uncomment) the property, set it to</td>
</tr>
<tr>
<td>true, and then restart the server.</td>
</tr>
<tr>
<td>Example (commented out):</td>
</tr>
<tr>
<td># account.restore.password.suppress=true</td>
</tr>
</tbody>
</table>

Create password check box properties

Create password check box properties define the default check box properties to create a password.

Table 60 defines the default create password check box properties.

Table 60. Create password check box default properties

<table>
<thead>
<tr>
<th>enrole.CreatePassword</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies whether a password is created automatically. Valid values are:</td>
</tr>
<tr>
<td>• true – Create a password.</td>
</tr>
<tr>
<td>• false – Do not create a password. The user must type in the password.</td>
</tr>
<tr>
<td>Example (default):</td>
</tr>
<tr>
<td>enrole.CreatePassword=true</td>
</tr>
</tbody>
</table>

Access catalog properties

The com.ibm.itim.accesscatalog.groupIntersectionJoin.enabled enables support for searching group access when requesting access in the Identity Service Center when Intersection Join directive is used for the group attribute. The com.ibm.itim.accesscatalog.customJoin.enabled enables support for searching group access when requesting access in the Identity Service Center when Custom Join directive is used for the group attribute.

Table 61. Access catalog properties

<table>
<thead>
<tr>
<th>com.ibm.itim.accesscatalog.groupIntersectionJoin.enabled</th>
</tr>
</thead>
</table>
Table 61. Access catalog properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| Do not change this property value unless you are a qualified administrator. Enables support for searching group access when requesting access in the Identity Service Center in the case where *Intersection Join directive* is used for the group attribute. Values include:  
  • `true`  
  • `false`  
The default is `false`. Example (default):  
  `com.ibm.itim.accesscatalog.groupIntersectionJoin.enabled=false` |

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| `com.ibm.itim.accesscatalog.customJoin.enabled` | Do not change this property value unless you are a qualified administrator. Enables support for searching group access when requesting access in the Identity Service Center in the case where *Custom Join directive* is used for the group attribute. Values include:  
  • `true`  
  • `false`  
The default is `false`. Example (default):  
  `com.ibm.itim.accesscatalog.customJoin.enabled=false` |

Identity feed properties

Identity feed properties define a default identity feed action, such as whether to suspend an account. Table 62 defines the default identity feed properties.

Table 62. Default identity feed properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| `enrole.suspend.accounts.identity.feed` | Specifies whether all of a user's accounts are suspended when the person is suspended during an identity feed. Valid values are:  
  • `true` – Suspend all accounts of a suspended user.  
  • `false` – Do not suspend all accounts of a suspended user.  
Example (default):  
  `enrole.suspend.accounts.identity.feed=true` |
Upgrade properties

Upgrade properties define values for the upgrade of a specific release of IBM Security Identity Manager.

Table 63 defines the product upgrade properties.

Table 63. Default upgrade properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>minUpgradeVersion</td>
<td>Specifies the minimum version that the upgrade supports for a specific release of IBM Security Identity Manager.</td>
<td>minUpgradeVersion=5.0</td>
</tr>
<tr>
<td>file.merge.list</td>
<td>Specifies which properties files are merged during the upgrade of IBM Security Identity Manager.</td>
<td>file.merge.list=enRole \ enRoleLDAPConnection \ enRoleDatabase \ enRoleLogging \ enRoleMail \ ui \ CustomLabels \ CustomLabels_en \ enRoleAuthentication \ adhocreporting \ enroleworkflow \ enroleAuditing \ SelfServiceScreenText \ SelfServiceScreenText_en \ SelfServiceHelp \ SelfServiceUI \ SelfServiceHomePage \ scriptframework \ encryptionKey \ KMIPServer</td>
</tr>
</tbody>
</table>

Back up these files with backupPropertyFiles.sh or backupPropertyFiles.cmd.

Multiple password-synch agent properties

Multiple password-synch agent properties are used to configure the IBM Security Identity Manager Server to support multiple password-synchronization agents.

Table 64 defines the properties used to configure the support for multiple password-synch agents.

Table 64. Multiple password-synch agent properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.passwordsynch.enabledonresource</td>
<td></td>
</tr>
</tbody>
</table>
Table 64. Multiple password-synch agent properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.passwordsynch.enabledonresource</td>
<td>Specifies whether to enable or disable the support for multiple password-synch agents. Valid values are:</td>
</tr>
<tr>
<td></td>
<td>• <strong>true</strong> – Enable the support for multiple password-synch agents</td>
</tr>
<tr>
<td></td>
<td>• <strong>false</strong> – Disable the support for multiple password-synch agents</td>
</tr>
<tr>
<td></td>
<td>Example (default):</td>
</tr>
<tr>
<td></td>
<td>enrole.passwordsynch.enabledonresource=false</td>
</tr>
<tr>
<td>enrole.passwordsynch.toleranceperiod</td>
<td>Specifies the maximum time duration, in <strong>seconds</strong>, between a password change request sent from the IBM Security Identity Manager Server to the password synch resource, and receiving a reverse password synch request from the plug-in installed on the password synch resource.</td>
</tr>
<tr>
<td></td>
<td>Example (default):</td>
</tr>
<tr>
<td></td>
<td>enrole.passwordsynch.toleranceperiod=60</td>
</tr>
<tr>
<td>enrole.PasswordSynchStoreMonitor.heartbeat</td>
<td>Specifies the password synch transaction monitor heartbeat, in <strong>hours</strong>.</td>
</tr>
<tr>
<td></td>
<td>Example (default):</td>
</tr>
<tr>
<td></td>
<td>enrole.PasswordSynchStoreMonitor.heartbeat=1</td>
</tr>
</tbody>
</table>
Notices

This information was developed for products and services offered in the U.S.A.

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 give 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 (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.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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 might 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
2Z4A/101
11400 Burnet Road
Austin, TX 78758 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 measurement 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 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. You may copy, modify, and distribute these sample programs in any form without payment to
Privacy Policy Considerations

This information was developed for products and services that are offered in the US and the European Union.

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

Index

A
access control 8
API 8
accessibility x, 1
account
object, JavaScript extension 72
account restore information 264
AccountModelExtension, JavaScript extensions 56
AccountSearch object, JavaScript extension 72
AccountSearch.searchByOwner object, JavaScript extension 72
AccountSearch.searchByUid object, JavaScript extension 73
AccountSearch.searchByURI object, JavaScript extension 73
activity object, JavaScript extension 75
Activity.auditEvent object, JavaScript extension 76
Activity.description object, JavaScript extension 77
Activity.duedate object, JavaScript extension 77
Activity.guid object, JavaScript extension 77
Activity.id object, JavaScript extension 78
Activity.index object, JavaScript extension 78
Activity.name object, JavaScript extension 78
Activity.participant object, JavaScript extension 78
Activity.resultDetail object, JavaScript extension 79
Activity.resultSummary object, JavaScript extension 79
Activity.setResult object, JavaScript extension 79
Activity.started object, JavaScript extension 80
Activity.state object, JavaScript extension 80
Activity.subtype object, JavaScript extension 80
Activity.type object, JavaScript extension 81
ad hoc reporting.properties 175
APIs (continued)
self registration 8
service provider 11
shared access APIs 12
single sign-on 14
web services API 15
workflow 18
APIs group
recertification policy 9
application client request
configuration 261
application extension methods 3
application server information 235
AttributeChangeOperation object, JavaScript extension 81
AttributeChangeOperation.attr object, JavaScript extension 81
AttributeChangeOperation.op object, JavaScript extension 81
AttributeChangeOperation.values, JavaScript extension 82
AttributesExtension, JavaScript extensions 54
authentication
API 10
authentication properties
enRoleAuthentication.properties 186
contextual parameters 167
parameter names 168
writing 169
create password checkbox information 265
credential
shared access module 88
Credential.getAccessMode() 90
Credential.isNotifyOnly() object, JavaScript extension 91
Credential.isPasswordViewable() object, JavaScript extension 92
Credential.isResetPasswordAtCheckin() object, JavaScript extension 92
CredentialForms.properties, not configurable 173
customization
date range 27
CustomLabels.properties supplemental properties 182

C
cache information 241
challenge response encoding information 249
ConfigErrorMessages.properties, not configurable 173
ConfigLabels.properties, not configurable 173
ConfigMessages.properties, not configurable 173
constructor
JavaScript migration, example 66
ContainerSearch object, JavaScript extension 82
ContainerSearch.searchByFilter object, JavaScript extension 83
ContainerSearch.searchByURI object, JavaScript extension 83
customization
date range 27
default notification templates
manual service 35
default recertification templates
recertification default messages 36
default workflow templates
workflow default messages 42
delegate
DelegateExtension, JavaScript extensions 55
dictionary
password policy 19
DirectoryName object, JavaScript extension 93
DirectoryName.addProperty object, JavaScript extension 94
DirectoryName.dn object, JavaScript extension 95
DirectoryName.getChanges object, JavaScript extension 95
DirectoryName.getProperty object, JavaScript extension 96

D
data services
API 7
database cleanup information 264
database resource bundle 264
DataBaseFunctions.conf 182
dataservices attributes
recertification 21
date range customization 27
default notification templates
manual service 35
default recertification templates
recertification default messages 36
default workflow templates
workflow default messages 42
delegate
DelegateExtension, JavaScript extensions 55
dictionary
password policy 19
DirectoryName object, JavaScript extension 93
DirectoryName.addProperty object, JavaScript extension 94
DirectoryName.dn object, JavaScript extension 95
DirectoryName.getChanges object, JavaScript extension 95
DirectoryName.getProperty object, JavaScript extension 96

© Copyright IBM Corp. 2012, 2013
JavaScript extension (continued)

- Credential.getCheckoutDuration() 90
- Credential.getNotificationRecipient() 90
- Credential.getNotifyOption() 90
- Credential.isCheckoutSearchEnable() 91
- Credential.isNotifyOnly() 91
- Credential.isPasswordViewable() 92
- Credential.isResetPasswordAtCheckin() 92
- DirectoryObject.getName() 97
- DirectoryObject.addProperty 94
- DirectoryObject.dn 95
- DirectoryObject.getChanges() 95
- DirectoryObject.getProperty 96
- DirectoryObject.name 98
- DirectoryObject/profileName 98
- EmailContext 100
- Enrole 102
- Enrole/generatePassword 103
- Enrole.getAttributeValue 103
- Enrole.getAttributeValues 104
- Enrole/localize 104
- Enrole/log 104
- Enrole/logError 105
- Enrole/loginInfo 106
- Enrole/Warning 106
- Enrole.toGeneralizedTime 107
- Enrole/toMilliseconds 107
- ExtendedPerson/ownerName 110
- ExtendedPerson/ownerType 110
- IdentityPolicy 113
- IdentityPolicy/nextCount 113
- IdentityPolicy/userIDExists 114
- PackagedApprovalDocument 114
- PackagedApprovalItem 116
- Participant 117
- Participant/implementation 118
- Participant/name 119
- Participant/type 119
- Participant/Type 119
- Person 121
- Person/definedRoleDN() 125
- Person/getNewRoles 125
- Person/getNewRoles() 125
- Person/getRoleAssignmentData() 123
- Person/getRoleAssignmentData() 123
- Person/roles 124
- Person/isInRole 125
- Person/removeRole 126
- Person/updateRoleAssignmentData() 125
- Person/updateRoleAssignmentData() 125
- PostOffice 129
- PostOffice/getAllEmailMessages() 129
- PostOffice/getEmailAddress 130
- PostOffice/getEmailAddress 130
- PostOffice/GetTopic 130
- Process 131
- Process/auditEvent 133
- Process/commit 133
- Process/description 133
- Process/getActivity 133
- Process/getParent 134
- Process/getRootProcess() 134
- Process/getRootRequesterName() 135
- Process/guid 135
- Process/id 135
- Process/Name 135
- Process/parentID 136
- Process/requesteeDN 136
- Process/requesteeName 136
- Process/requestorDN 136
- Process/requestorName 137
- Process/requestorType 137
- Process/resultDetail 137
- Process/resultSummary 138
- Process/setResult 138
- Process/setSubjectData 139
- Process/setSubjectData 139
- Process/state 139
- Process/subject 140
- Process/type 140
- Process/Data 140
- Process/Data/Get 141
- Process/Data/Set 141
- RecertificationWorkflow 142
- Reminder 142
- Role 143
- Role/getAssignmentAttributes 144
- Role/Owner 145
- Role/setAssignmentAttributes 145
- Role/Search 152
- Role/Search/searchByName 153
- Role/Search/searchByURI 153
- Service/Search 155
- objects 69
- Role/getAllAssignmentAttributes 144
- Role/Assignment/addProperty 150
- Role/Assignment/getAttributeValues 150
- Role/Assignment/GetChanges() 150
- Role/Assignment/GetDefinedRoleDN() 149
- Role/Assignment/GetProperty 151
- Role/Assignment/GetPropertyNames 151
- Role/Assignment/removeProperty 152
- Role/Assignment/setProperty 152
- Service/Search/searchByFilter 155
- Service/Search/searchByName 156
JavaScript objects...

JavaScript functions...

JavaScript extension (continued)
ServiceSearch.searchByFilter 83
ServiceSearch.searchByURI 83
account 72
AccountSearch 72
AccountSearch.searchByUid 73
AccountSearch.searchByURI 74
activity 75
Activity.auditEvent 76
Activity.description 77
Activity.duedate 77
Activity.guid 77
Activity.id 78
Activity.index 78
Activity.name 78
Activity.participant 78
Activity.resultDetail 79
Activity.resultSummary 79
Activity.setResult 79
Activity.started 80
Activity.state 80
Activity subtype 80
Activity type 81
AttributeChangeOperation 81
AttributeChangeOperation.attr 81
AttributeChangeOperation.op 82
ContainerSearch 82
ContainerSearch.searchByFilter 83
ContainerSearch.searchByURI 83
Context 84
Context.getAccountId 85
Context.getActivityResult 86
Context.getActivityResultByld 86
Context.getLoopCount 86
Context.getLoopCountById 87
Context.getProcessType 87
Context.getRequestee 87
Context.getService 88
Credential.getCheckoutDuration() 90
Credential.getNotificationRecipient() 90
Credential.getNotifyOption() 90

object (continued)

JavaScript extension (continued)

Credential.isCheckoutSearchEnable() 91
Credential.isNotifyOnly() 91
Credential.isPasswordViewable() 92
Credential.isResetPasswordAtCheckin() 92
DirectoryObject 93
DirectoryObject.getPartitionName() 97
DirectoryObject.appendProperty() 97
DirectoryObject.removeProperty ,
JavaScript extension 98
DirectoryObject.setProperty, 98
DirectoryObject.getPropertyAsDate 97
DirectoryObject.getPropertyAsString 97
DirectoryObject.getProperty() 96
DirectoryObject.getName() 98
DirectoryObject.profile() 98
DirectoryObject.setObject() 98

L
Labels.properties, not configurable 173
LDAP connection pool information 245
LDAP server information 237
life cycle rule 261
Log4j 194
LoopCountExtension, JavaScript
extensions 56

M
mail services configuration 250
mail templates
examples 34
manual service
default notification templates 35
Messages.properties, not configurable 173
messaging information 242
methods
RoleAssignmentObject 148

N
notices 269
null types 161

O
object 82, 90
Context.isAccountDataChanged object,
JavaScript extension 88
delegate JavaScript extension 92
EntityType 92
DirectoryObject.getPropertyAsDate 97
DirectoryObject.getPropertyAsString 97
DirectoryObject.removeProperty ,
JavaScript extension 98
DirectoryObject.setProperty object,
JavaScript extension 99
DirectoryObject.removeProperty ,
JavaScript extension 99
JavaScript extension
account 72
AccountSearch 72
AccountSearch.searchByUid 73
AccountSearch.searchByURI 74
activity 75
Activity.auditEvent 76
Activity.description 77
Activity.duedate 77
Activity.guid 77
Activity.id 78
Activity.index 78
Activity.name 78
Activity.participant 78
Activity.resultDetail 79
Activity.resultSummary 79
Activity.setResult 79
Activity.started 80
Activity.state 80
Activity subtype 80
Activity type 81
AttributeChangeOperation 81
AttributeChangeOperation.attr 81
AttributeChangeOperation.op 82
ContainerSearch 82
ContainerSearch.searchByFilter 83
ContainerSearch.searchByURI 83
Context 84
Context.getAccountId 85
Context.getActivityResult 86
Context.getActivityResultByld 86
Context.getLoopCount 86
Context.getLoopCountById 87
Context.getProcessType 87
Context.getRequestee 87
Context.getService 88
Credential.getCheckoutDuration() 90
Credential.getNotificationRecipient() 90
Credential.getNotifyOption() 90

object (continued)

JavaScript extension (continued)

Credential.isCheckoutSearchEnable() 91
Credential.isNotifyOnly() 91
Credential.isPasswordViewable() 92
Credential.isResetPasswordAtCheckin() 92
DirectoryObject 93
DirectoryObject.
getPropertyNames 97
DirectoryObject.appendProperty() 97
DirectoryObject.removeProperty ,
JavaScript extension 98
DirectoryObject.setProperty, 98
DirectoryObject.getPropertyAsDate 97
DirectoryObject.getPropertyAsString 97
DirectoryObject.getProperty() 96
DirectoryObject.getName() 98
DirectoryObject.profile() 98
DirectoryObject.setObject() 98

K
keyboard navigation 1

L
Labels.properties, not configurable 173
LDAP connection pool information 245
LDAP server information 237
life cycle rule 261
Log4j 194
LoopCountExtension, JavaScript
extensions 56
RoleAssignmentObject methods 148
RoleAssignmentObject.getAssignedRoleDN() Javascript extension 149
RoleAssignmentObject.getChanges() Javascript extension 150
RoleAssignmentObject.getDefinedRoleDN() Javascript extension 149
RoleAssignmentObject.getProperty object Javascript extension 151
RoleAssignmentObject.getPropertyNames object Javascript extension 151
RoleAssignmentObject.removeProperty object Javascript extension 152
RoleAssignmentObject.setProperty object Javascript extension 152
RoleModelExtension, Javascript extensions 57
RoleSearch object, Javascript extension 152
RoleSearch.searchByName object, Javascript extension 153
RoleSearch.searchByURI object, Javascript extension 153

Scheduling information 244
script service selection policy 165
scriptframework.properties 61, 215
JavaScipt configuring 62
search strategy and LDAP control configuration 283
SelfServiceHelp.properties 217
SelfServiceHomePage.properties 217
SelfServiceScreenText.properties 218
SelfServiceUI.properties 218
SeparationOfDutyRuleViolation object Javascript extension 154
service service selection policy Javascript 165
service object, Javascript extension 154
service selection policy Javascript objects 165
script 165
ServiceExtension, Javascript extensions 59
ServiceModelExtension, Javascript extensions 57
ServiceSearch object, Javascript extension 155
ServiceSearch.searchByFilter object Javascript extension 155
ServiceSearch.searchByName object Javascript extension 156
ServiceSearch.searchByName object Javascript extension 156
ServiceSearch.searchByURI object Javascript extension 156
ServiceSearch.searchForClosestToPerson object Javascript extension 157
shared access (continued)
JavaScript APIs 14
shared access web services API 14
shared access application APIs 14
shared access data model APIs 12
shared access module credential 88
shared secret hashing 260
SubForm control type 167
template contextual parameters 167
parameter names 168
writing 169
subform, properties, not configurable 173
SubjectExtension, Javascript extensions 59
supplemental properties additional, not configurable 173
ConfigLabels.properties 173
CustomForms.properties 173
Dsml2RootDSE.properties 173
Dsml2Schema.properties 173
enRole2Idif.properties 173
enRoleHelp.properties 173
enRoleInstaller.properties 173
Labels.properties 173
Messages.properties 173
passwordrules.properties 173
platformcontext.properties 173
Properties.properties 173
subform.properties 173
tenant.properties 173
tmsMessages.properties 173
adhocreporting.properties 175
CustomLabels.properties 182
DataBaseFunctions.conf 182
enroleAuditing.properties 183
enroleDatabase.properties 188
enRoleLDAPConnection.properties 191
enRoleLogging.properties 194
enRoleMail.properties 205
enrolepolicies.properties 208
enroleworkflow.properties 211
extensions.properties 212
helpmappings.properties 214
properties files 171
reportingLabels.properties 214
reporttabledeny.properties 214
ReversePasswordSynchronization 262
Role object, Javascript extension 143
Role.setAllAssignmentAttributes object Javascript extension 144
Role.getAssignmentAttributes object, Javascript extension 144
Role.getOwner object, Javascript extension 145
Role.setAssignmentAttributes object, Javascript extension 145
RoleAssignment.addProperty object Javascript extension 150
RoleAssignmentAttribute getRoleName() 146
RoleAssignmentAttribute object, Javascript extension 146
RoleAssignmentAttribute.getName() Javascript extension 146
RoleAssignmentAttribute.getRoleDN Javascript extension 147
shared access (continued)
authorization extension API 14
system properties (continued)
database resource bundle 264
encryption information 247
enRole.properties file 229
identity feed 266
LDAP connection pool
information 245
LDAP server information 237
life cycle rule 261
mail services configuration 250
messaging information 242
organization name 236
password synchronization 267
password transaction monitor
settings 244
person profile 240
post office 262
product name 261
properties files 171
reconciliation information 257
remote services properties files 233
reverse password
synchronization 262
scheduling information 244
search strategy and LDAP control
configuration 238
shared secret hashing 260
system configuration program 249
tenant information, default 236
understanding properties files 229
upgrade 267
web services properties files 233
WebSphere-specific configuration 229
workflow configuration
information 250
XML and DTD information 245

WorkflowExtension, JavaScript
extensions 59
workflows
application extension
methods 3
registering 4
JavaScript objects
relevant data 60

X
XML and DTD information 245

T
tenant information, default 236
tenant.properties, not configurable 173
terminology ix
tmsMessages.properties, not
configurable 173
training x
troubleshooting x
TungstenTheme.properties, not
configurable 173

U
ui.properties 221
upgrade information 267

W
WebSphere-specific configuration 229
workflow
application extensions 3
workflowApplication interface 3
workflow configuration information 250
workflow default messages
default workflow templates 42
workflow extensions
recertification 23

Index 279