Version 2 Release 2

IBM i2 Enterprise Insight Analysis Configuring a deployment before going live

Note

Before using this information and the product it supports, read the information in "Notices" on page 157.

This edition applies to version 2, release 2, modification 0 of IBM® i2® Enterprise Insight Analysis (product number 5725-G23) and to all subsequent releases and modifications until otherwise indicated in new editions. Ensure that you are reading the appropriate document for the version of the product that you are using. To find a specific version of this document, access the Configuring section of the IBM Knowledge Center, and ensure that you select the correct version.

Contents

Going live to Enterprise Insight Analysis users

After you successfully create an example deployment of i2 Enterprise Insight Analysis, the next phase is to prepare it for production by customizing it for your organization. Before your deployment goes live, you must provide it with an i2 Analyze schema and a security schema, and you might decide to add services or data stores to the base functionality.

Modifying the basic deployment

In the deployment toolkit, you can identify and modify the core components of your deployment. You can use the toolkit to customize aspects of your deployment.

Modifying the environment properties

The script in the deployment toolkit that deploys i2 Analyze requires information about the deployment environment. In the environment properties you can identify your DB2 location, specify the installation directories that i2 Analyze uses, and specify the locations that i2 Analyze can store data.

About this task

Each application that is defined in your topology file has an associated folder in the `toolkit\configuration\environment` directory that stores information about the environment that is specific to that application. There are two main files that are stored for each of your applications:

`environment.properties`

This file contains the file paths to use with the application. All the properties in this file must have values that match the environment.

`environment-advanced.properties`

This file contains additional options about your environment. Although you can modify the values in this file, you can deploy the application without making any changes.

The following properties are in the `environment.properties` file:

`db.installation.dir.db2`

The installation path of DB2.

`db.database.location.dir.db2`

The path to the directory at the root of DB2 database file storage.

`wlp.home.dir`

The installation path for WebSphere Application Server Liberty profile.

Important: On Linux, when you run the deployment script, ensure that you have permission to read, write, and execute files in these directories.

`java.home.dir`

The installation path for the IBM Java JDK.

`solr.home.dir`

The installation path for Apache Solr and ZooKeeper.

Note: This property is only required in a deployment that uses the Opal services.

apollo.data

The path to a directory where i2 Analyze temporary files are stored.

Procedure

For each application:

1. Using a text editor, open the `environment.properties` file.
2. Using the preceding descriptions, ensure that the property values match the deployment environment.
3. Save and close the file.

What to do next

After you update the environment properties, you can either modify other aspects of the deployment toolkit or redeploy the system to update the deployment with any changes. For more information about redeploying your system, see "Deploying i2 Analyze" on page 150.

Modifying the topology

The `topology.xml` file contains the description of the physical architecture of the system that you want to deploy. To change the physical architecture of the system, you must update the `topology.xml` file and redeploy the system.

About this task

The `topology.xml` file contains the information that the `setup` script requires to identify the prerequisites that are being used, where the system stores information, and the fragments to combine to create the application.

You can make a number of changes to the `topology.xml` file. For more information about the changes that you can make in the `topology.xml` file, see "The topology" on page 2.

Procedure

1. Using an XML editor, open `toolkit\configuration\environment\topology.xml`.
2. Change the values of any attributes to match your environment.
3. Check that the locations that i2 Analyze can store files match your environment.

   Note: Ensure that any paths you set are to locations where the application server has permission to store files.
4. Save your changes and close `topology.xml`.

What to do next

After you update the topology file, you can either modify other aspects of the deployment toolkit or redeploy the system to update the deployment with any changes. For more information about redeploying your system, see "Deploying i2 Analyze" on page 150.

The topology

In the deployment toolkit, the topology file defines the physical architecture of the finished deployment, and the components that make up the pieces of that architecture. The topology file contains domain-specific information that you must provide before deployment can take place.

The `topology.xml` file contains distinct sections for defining different aspects of a deployment.

`i2-data-sources`

A definition of each data source that is available within the deployment.

`databases`

The names, types, and locations of the databases that i2 Analyze uses to store data.

`applications`

The applications that comprise this deployment of i2 Analyze, and the locations of the application servers on which they are to be installed.

`zookeepers` and `solr-clusters`

The ZooKeeper hosts and Solr clusters that are used in this deployment.

`connectors`

The connectors that are used with Connect in this deployment.

Data sources

The topology files that are supplied with each example in the deployment toolkit contain preconfigured `<i2-data-source>` elements. You can add additional `<i2-data-source>` elements to define other data sources.

The `<i2-data-sources>` element must contain an `<i2-data-source>` element for each data source that the deployment connects to. For example:

```
<i2-data-sources>
    <i2-data-source ar="false" default="false" id="infostore">
        <DataSource Version="0" Id="">
            <Shape>InfoStore</Shape>
            <Name>Information Store</Name>
        </DataSource>
     </i2-data-source>
</i2-data-sources>
```

Where:

Attribute	Description
`id`	A unique identifier that is used to distinguish this data source throughout the system.
`ar`	Indicates whether this data source is the Analysis Repository. This attribute must be set to `true` for the Analysis Repository, and to `false` for any other data source.
`default`	An optional attribute that is used to specify that a different data source from the Analysis Repository should be marked as the default searching option within the Intelligence Portal. To specify a different data source as the default, set this attribute to `true` for that data source.

Each `<i2-data-source>` contains a single `<DataSource>` element that has two standard attributes and two child elements.

Attribute	Description
`Id`	Reserved for future use. The value must be empty in this version of i2 Analyze.
`Version`	Reserved for future use. The value must be `0` in this version of i2 Analyze.

Element	Description
`Shape`	Specifies the type of structure that the data source uses. You can specify one of the following values: • `InfoStore` • `Repository` • `DAOD`
`Name`	The name of this data source, which is presented to users.

In addition, for data sources that you connect to using data access on-demand, you must also specify the following attributes on the `<DataSource>` element:

Attribute	Description
`EdrsPresent`	Indicates whether this data source has an external data retrieval service.
`EdrsGetContextSupported`	Indicates whether this data source has an external data retrieval service that supports "get context" operations.
`EdrsGetLatestItemsSupported`	Indicates whether this data source has an external data retrieval service that supports "get latest items" operations.
`SesPresent`	Indicates whether this data source has a subset exploration service.
`ScsPresent`	Indicates whether this data source has a subset creation service.
`ScsBrowseSupported`	Indicates whether this data source has a subset creation service that supports "browse" (that is, unbound search) operations.
`ScsSearchSupported`	Indicates whether this data source has a subset creation service that supports "search" operations.
`ScsNetworkSearchSupported`	Indicates whether this data source has a subset creation service that supports "network search" operations.
`ScsDumbbellSearchSupported`	Indicates whether this data source has a subset creation service that supports "dumbbell search" operations.
`ScsFilteredSearchSupported`	Indicates whether this data source has a subset creation service that supports "filtered search" operations.

Databases

The `<databases>` element defines each database that i2 Analyze uses to store data. The `<database>` elements contain attributes that define information about the database, and the mechanism that is used to connect to it.

For example:

```
<database database-type="InfoStore"
dialect="db2" database-name="ISTORE" instance-name="DB""
xa="false" edition="" version="" host-name="host" id="infostore"
port-number="50000" />
```

Where:

Attribute	Description
`database-type`	Identifies the type of the database within i2 Analyze. This can be one of the following values: • InfoStore • WriteStore • ccConfig
`dialect`	Specifies the type of database engine. This attribute can be set to one of the following values: • `db2` • `sqlserver` • `oracle`
`database-name`	A name that identifies the database to the database engine.
`instance-name`	The instance name that was specified during installation of your database engine.
`xa`	Determines whether distributed transactions are enabled for this database.
`edition`	Identifies the DB2 edition. This can be one of the following values: • db2aese • db2awse • db2wse • db2ese • db2exp • db2expc • db2consv
`version`	Identifies the DB2 version number.
`host-name`	The host name of the server where the database is located.
`id`	A unique identifier that is used to distinguish this database throughout the system.
`port-number`	The port on the server to which to send requests for this database.

Attribute	Description
max-pool-size	Sets the maximum number of connections that are allowed in the connection pool.

Applications

The <applications> element within the topology file describes the location and structure of the i2 Analyze applications. Each application to deploy on the application server is described in the child <application> element.

For each application in the deployment, the topology file contains an <application> element that defines the WAR file and any indexes for that application.

Indexes

The <lucene-indexes> element defines the Lucene indexes to be used by the onyx-server application.

At a minimum, the <lucene-indexes> element for the onyx-server application requires a <lucene-index> child element for the Analysis Repository index.

Attribute	Description
id	A unique identifier that is used to distinguish this Lucene index.
main-index-location	The location that is used to store the main index.
alternatives-location	The location that is used to store the alternative term index.

File stores

The <file-stores> element defines the file stores that are used by the opal-server application. The location attribute specifies the file path to use for each file store. The other attributes must be left with their default values.

WAR files

The <wars> element contains child <war> elements that define the contents of the i2 Analyze WAR files that are installed on the application server.

Each <war> element has the following attributes:

Attribute	Description
target	The type of WAR file to create. The following types are available: • onyx-services-ar • opal-services-is • onyx-services-daod • opal-services-daod • iBase • connectorCreator
name	The name of the WAR file. By convention, the deployment scripts expect a directory with the same name to be present.
i2-data-source-id	A name that identifies the <i2-data-source> element for this WAR file.

Attribute	Description
	For the onyx-services-ar WAR, this must reference a data source of shape Repository, and for the opal-services-is WAR, this must reference a data source of shape InfoStore.
context-root	The URL that is used to access the WAR file. For the onyx-services-ar WAR, this is apollo by default. For the opal-services-is WAR, this is opal by default. For the opal-services-daod WAR, this is opaldaod by default.

In addition, the following child elements can be defined:

data-sources

Identifies each database that is used by this application in a separate <data-source> element. Each <data-source> element has the following attributes:

Attribute	Description
database-id	Identifies the database to use. This value must match the id value that is specified in a <database> element. For the onyx-services-ar WAR, one <data-source> element must reference a database of type WriteStore, and for the opal-services-is WAR, one must reference a database of type InfoStore.
create-database	Determines whether the database is to be created on the server that is specified in the <database> element. Set as *true* to create the database, or *false* to not create the database.

lucene-index-ids

Identifies the Lucene indexes that are used by this WAR in a child <lucene-index-id> element. Each Lucene index is identified by the value attribute, the value of which must match an id value that is specified in a <lucene-index> element.

solr-collection-ids

Identifies the Solr collection that is used by this application. The Solr collection is identified by the collection-id attribute, the value of which must be main_index and match the id value that is specified in a <solr-collection> element. A Solr collection belongs to a Solr cluster. Each Solr cluster is identified by the cluster-id attribute, the value of which must match an id value that is specified in a <solr-cluster> element.

The Solr collection requires a directory to store data before it is added to the Solr index, the value of the data-dir attribute specifies the file path for the directory.

For more information about the ZooKeeper and Solr related elements in the topology.xml file, see “ZooKeeper and Solr” on page 9.

file-store-ids

Identifies the file stores that are used by this application. Each file store is identified by the value attribute, the value of which must match an id value that is specified in a <file-store> element.

connector-ids

Identifies the connectors that are available in the `opal-services-daod` WAR in a child `<connector-id>` element. Each connector is identified by the `value` attribute, that value of which must match an `id` value that is specified in a `<connector>` element.

For more information about connector related elements in the `topology.xml` file, see "Connectors" on page 11.

fragments

Specifies the fragments that are combined to create the application.

Note: All applications must contain the `common` fragment.

In the supplied `topology.xml` file for the Analysis Repository example deployment, the `onyx-server` application definition is:

```
<application host-name="" http-server-host="true" name="onyx-server">
  <lucene-indexes>
    <lucene-index id="ar" main-index-location=""
      alternatives-location=""/>
  </lucene-indexes>
  <wars>
    <war context-root="apollo" i2-data-source-id="ar-id" name="onyx-server-ar"
    target="onyx-services-ar">
      <data-sources>
        <data-source create-database="true" database-id="write1"/>
      </data-sources>
      <lucene-index-ids>
        <lucene-index-id value="ar"/>
      </lucene-index-ids>
      <fragments>
        <fragment name="common"/>
        <fragment name="onyx-services-ar"/>
      </fragments>
    </war>
  </wars>
</application>
```

In the supplied `topology.xml` file for the Information Store Opal example deployment, the `opal-server` application definition is:

```
<application http-server-host="true" name="opal-server" host-name="">
  <wars>
    <war context-root="opal" name="opal-services-is" i2-data-source-id="infostore"
     target="opal-services-is">
      <data-sources>
        <data-source database-id="infostore" create-database="true" />
      </data-sources>
      <file-store-ids>
        <file-store-id value="chart-store" />
        <file-store-id value="job-store" />
        <file-store-id value="recordgroup-store" />
      </file-store-ids>
      <fragments>
        <fragment name="opal-services-is" />
        <fragment name="common" />
        <fragment name="default-user-profile-provider" />
      </fragments>
      <solr-collection-ids>
        <solr-collection-id collection-id="main_index" data-dir=""
         cluster-id="is_cluster" />
      </solr-collection-ids>
    </war>
  </wars>
  <file-stores>
    <file-store location="" id="chart-store" type="chart-store" />
    <file-store location="" id="job-store" type="job-store" />
    <file-store location="" id="recordgroup-store" type="recordgroup-store" />
  </file-stores>
</application>
```

Note: If you set the `http-server-host` attribute of the `<application>` element to false, the deployment toolkit does not configure the HTTP Server.

ZooKeeper and Solr

The Opal services use Apache Solr for the text indexing and search capabilities. ZooKeeper is the service that is used to maintain configuration information and distributed synchronization across Solr.

The `topology.xml` file for a deployment that includes the `opal-server` application, also includes the `<zookeepers>` and `<solr-clusters>` elements. The `<solr-clusters>` and `<zookeepers>` elements define the Solr clusters that are used in a deployment and the ZooKeeper instance that manages them.

`<zookeepers>`

The `<zookeepers>` element includes one or more child `<zookeeper>` elements. The `id` attribute of the `<zookeeper>` element is a unique identifier for the ZooKeeper instance. To associate the ZooKeeper instance with a Solr cluster, the value of the `id` attribute must match the value of the `zookeeper-id` attribute of a `<solr-cluster>` element.

The <zkhosts> element is a child of the <zookeeper> element. The <zkhosts> element can have one or more child <zkhost> elements. Each <zkhost> element has the following attributes:

Attribute	Description
id	A unique identifier that is used to identify the ZooKeeper host. This value must be an integer between 1 - 255.
host-name	The host name of the ZooKeeper host.
data-dir	The location that ZooKeeper uses to store data.
port-number	The port number of the ZooKeeper host.

In the supplied topology.xml file that includes the opal-server application, the <zookeepers> definition is:

```
<zookeepers>
  <zookeeper id="zoo">
    <zkhosts>
      <zkhost id="1" host-name="" data-dir="" port-number="9983" />
    </zkhosts>
  </zookeeper>
</zookeepers>
```

<solr-clusters>

The <solr-clusters> element includes one or more child <solr-cluster> elements. The id attribute of the <solr-cluster> element is a unique identifier for the Solr cluster. To associate the Solr cluster with a ZooKeeper instance, the value of the zookeeper-id attribute must match the value of the id attribute of a <zookeeper> element.

<solr-collections>

The <solr-collections> element is a child of the <solr-cluster> element. The <solr-collections> element can have one child <solr-collection> element. The <solr-collection> element has the following attributes:

Attribute	Description
id	The value must be set to main_index.
num-shards	The number of logical shards that are created as part of the Solr collection.
num-replicas	The number of physical replicas that are created for each logical shard in the Solr collection.
max-shards-per-node	The maximum number of shards that are allowed on each Solr node. This value is the result of *num-shards* multiplied by *num-replicas*.

<solr-nodes>

The <solr-nodes> element is a child of the <solr-cluster> element. The <solr-nodes> element can have one or more child <solr-node> elements. Each <solr-node> element has the following attributes:

Attribute	Description
id	A unique identifier that is used to identify the Solr node.
memory	The amount of memory that can be used by the Solr node.
host-name	The host name of the Solr node.
data-dir	The location that Solr stores the index.
port-number	The port number of the Solr node.

In the supplied topology.xml file that includes the opal-server application, the <solr-clusters> definition is:

```
<solr-clusters>
  <solr-cluster id="is_cluster" zookeeper-id="zoo">
    <solr-collections>
      <solr-collection id="main_index" max-shards-per-node="4" num-shards="4"
       num-replicas="1" />
    </solr-collections>
    <solr-nodes>
      <solr-node memory="2g" id="node1" host-name="" data-dir=""
       port-number="8983" />
    </solr-nodes>
  </solr-cluster>
</solr-clusters>
```

Connectors

An i2 Analyze Opal deployment that uses the i2 Connect services enables you to connect to external data sources. i2 Connect enables analysts to search for and retrieve data from external data sources and analyze the results on a chart.

The topology.xml file for a deployment that includes the opal-server application with the opal-services-daod war, also includes the <connectors> element. The <connectors> element defines the connectors that are used in a deployment.

<connectors>

The <connectors> element includes one or more child <connector> elements. The <connector> element can have the following attributes:

Attribute	Description
id	A unique identifier that is used to identify the connector.
name	The name of the connector, which is presented to users.
base-url	The URL to the connector, made up of host name and port number. For example, https://*host name*:*port number*. You can use the HTTP or HTTPS protocol.
configuration-url	The URL to any configuration that is required for the connector. The default value for the configuration-url attribute is /config.

Attribute	Description
	The presence of the attribute in the `topology.xml` file is optional. If it is not present, the default value is used.

In the supplied `topology.xml` file that includes the `opal-services-daod` war, the `<connectors>` definition is:

```
<connectors>
    <connector id="example-connector" name="Example"
      base-url="http://localhost:3700/" />
</connector>
```

Modifying the IBM HTTP Server properties

If the environment that you are deploying into has an HTTP server installation that is in a different location from the example deployment, you can specify the different locations. The values that describe your HTTP server installation are stored in the `http-server.properties` file, you can modify the values to match the deployment environment.

About this task

The property values in `http-server.properties` that specify the HTTP server directories are set to default values when the example is deployed. If you plan to use different locations for your HTTP server directories, modify the values in `toolkit\configuration\http-server.properties`. The following properties are available:

`http.server.home.dir`

The path to the installation directory of IBM HTTP Server. By default, `C:/IBM/HTTPServer` on Windows, or `/opt/IBM/HTTPServer` on Linux.

Important: On Linux, when you run the deployment script, ensure that you have permission to read, write, and execute files in the directories that IBM HTTP Server is installed.

`http.was.module.dir`

The path to the Web Server Plug-ins for WebSphere Application Server. By default, `C:/IBM/WebSphere/Plugins/bin/32bits` on Windows, or `/opt/IBM/WebSphere/Plugins/bin/64bits` on Linux.

Procedure

1. Using a text editor, open the `http-server.properties` file. You can find this file in the following location: `toolkit\configuration\environment`.
2. Using the preceding descriptions, ensure that the property values match the deployment environment.
3. Save and close the file.

What to do next

After you update the HTTP Server properties, you can either modify other aspects of the deployment toolkit or redeploy the system to update the deployment with any changes. For more information about redeploying your system, see “Deploying i2 Analyze” on page 150.

Modifying the database configuration

The example deployment configurations specify a local version of IBM DB2 as the database management system. When you set up your own system, you can deploy the Information Store and Analysis Repository on different servers, or for your instance of the Analysis Repository you can change the type of database management system.

Configuring remote IBM DB2 database storage

You can deploy i2 Analyze with DB2 database storage that is remote from the i2 Analyze server. When i2 Analyze is configured to deploy to a remote instance of DB2, the databases must be created and updated remotely without the i2 Analyze deployment toolkit present on the server that hosts the DB2 instance.

Before you begin

To deploy i2 Analyze using remote DB2 database storage, you must install DB2 on your database server, and DB2 or IBM Data Server Client on the application server. Both instances of DB2 must be installed according to the specifications defined in the i2 Analyze software prerequisites. For more information about installing the prerequisites, see Software prerequisites.

If you are using an existing installation of DB2 on the application server, you must remove any existing i2 Analyze databases. For more information about removing i2 Analyze databases, see “Removing databases from the system” on page 149.

About this task

To deploy with remote storage, the deployment toolkit must contain certain information about your DB2 instances. On the application server, update the deployment toolkit with the information about remote IBM DB2 instances. After you update the configuration, the specified databases are created and updated on the remote servers when you redeploy the application.

Note: You can complete the steps performed by the `catalogRemoteDB2Nodes` task manually. For example, if you are deploying a system that uses Transport Layer Security (TLS).

To catalog the remote nodes manually, you can run the `CATALOG TCPIP NODE` instead of using the `setup -t catalogRemoteDB2Nodes` command. For more information about the command, see `CATALOG TCPIP/TCPIP4/TCPIP6 NODE` command.

The following table shows how the `CATALOG` command parameters map to the values in the `topology.xml` file:

`CATALOG TCPIP NODE` command parameters	**`<database>` element attributes**
TCPIP NODE *nodename*	`node-name`
REMOTE *hostname*	`host-name`
SERVER *port number*	`port-number`
REMOTE_INSTANCE *instance-name*	`instance-name`

Procedure

1. Edit the `topology.xml` file to specify your remote DB2 databases:
   a) Using an XML editor, open `toolkit\configuration\environment\topology.xml`.
   b) Update the `host-name` and `port-number` attribute values of the `<database>` element to match the values of your remote DB2 instance.

Note: The value of the `instance-name` attribute must match the instance name of local instance DB2.

c) Add the `node-name` attribute to the `<database>` element of the databases to be hosted remotely. For example:

```
<database dialect="db2" xa="false" instance-name="DB2"
    database-name="WriteSto" database-type="WriteStore" id="write1"
    host-name="hostname" port-number="50000" node-name="node1" />
```

Where the value for `node-name` is the name of the node to create in the DB2 node directory. The value of the `node-name` attribute must start with a letter, and have fewer than 8 characters. For more information about naming in DB2, see Naming conventions.

Note: If the Analysis Repository and Information Store are using the same DB2 instance, they can use the same node.

d) If you are deploying the Information Store database, add the `os-type` attribute to the `<database>` element for the Information Store database. The value of the `os-type` is used to support the search functionality for the Information Store.
For example:

```
<database database-type="InfoStore" dialect="db2" instance-name="DB2"
     database-name="ISTORE" xa="false" edition="db2awse" id="infostore"
     host-name="hostname" port-number="50000" version="10.5"
     node-name="node1" os-type="WIN" />
```

Where the value for `os-type` is the operating system of the remote DB2 server.

Note: The value of the `os-type` attribute must be one of the following values: `AIX`, `UNIX`, or `WIN`.

Note: The values of the `edition` and `version` attributes must be correct for your remote instance of DB2. For more information about these attributes, see “Databases” on page 5.

2. Edit the `environment.properties` file, to specify the details of your remote and local instance of DB2.
   a) Using a text editor, open `toolkit\configuration\environment\server-name\environment.properties`.

   Where *server-name* is the name of your application server.

   b) Ensure that the value of the `db.installation.dir.db2` property is set for the local instance of DB2 on the application server.

   c) Set the value of the `db.database.location.dir.db2` property for the remote instance of DB2 on the database server.
3. Ensure that the users that are specified for your databases in the `toolkit\configuration\environment\credentials.properties` file are valid for your remote instance of DB2.
4. Run the following command to add the nodes that are defined in the `topology.xml` file to the DB2 node directory:

```
setup -t catalogRemoteDB2Nodes
```

Note: The directory cache is refreshed as part of this process without any further action.

5. If the Information Store or Analysis Repository databases that you are connecting to exist, you must catalog the databases against the remote nodes that you created. To catalog your remote DB2 databases, run the following command:

```
setup -t catalogDB2Databases
```

Note: The directory cache is refreshed as part of this process without any further action.

Results

A remote node is created with the name that is specified for the `node-name` attribute, and the database is cataloged against that node.

To check that the remote nodes and databases are cataloged, you can use the `listDB2NodeDirectory` and `listDB2SystemDatabaseDirectory` tasks:

- The `listDB2NodeDirectory` task lists the contents of the DB2 node directory.
- The `listDB2SystemDatabaseDirectory` task lists the contents of the local DB2 system database directory.

What to do next

Redeploy i2 Analyze so that the configuration changes are deployed to the application. For more information about redeploying your system, see "Deploying i2 Analyze" on page 150.

Modifying the remote database configuration

You can change the configuration attributes of your remote DB2 databases in the i2 Analyze deployment toolkit. To modify the remote database configuration, you must recatalog the remote nodes.

About this task

When you recatalog a remote node, the existing node is removed from the DB2 node directory and is then cataloged again with the updated configuration values.

If you need to configure security settings on the connection, you can recatalog the remote nodes manually, instead of using the tasks that are in the deployment toolkit. To recatalog the remote nodes manually, you can use the DB2 `UNCATALOG NODE` and `CATALOG TCPIP/TCPIP4/TCPIP6 NODE` commands.

You run the `UNCATALOG NODE` and `CATALOG NODE` commands manually instead of using the `recatalogRemoteDB2Nodes` task. For more information about the commands, see `UNCATALOG NODE` command and `CATALOG TCPIP/TCPIP4/TCPIP6 NODE` command.

Note: The `catalogRemoteDB2Nodes` and `recatalogRemoteDB2Nodes` tasks use the DB2 `CATALOG TCPIP NODE` command. The following table shows how the `CATALOG` command parameters map to the values in the `topology.xml` file:

`CATALOG TCPIP NODE` command parameters	**`<database>` element attributes**
TCPIP NODE *nodename*	`node-name`
REMOTE *hostname*	`host-name`
SERVER *port number*	`port-number`
REMOTE_INSTANCE *instance-name*	`instance-name`

Procedure

1. Change the host name, port, and operating system type attributes of the remote database:
   a) Using an XML editor, open `toolkit\configuration\environment\topology.xml`.
   b) Update the `host-name` and `port-number` attribute values of the `<database>` element for the database.
   c) Run the following command to recatalog the remote node:

```
setup -t recatalogRemoteDB2Nodes
```

2. Change the node name attribute of the remote database:
   a) Run the following command to uncatalog the remote node:

```
setup -t uncatalogRemoteDB2Nodes
```

   b) Using an XML editor, open `toolkit\configuration\environment\topology.xml`.
   c) Update the `node-name` attribute values of the `<database>` element for the database.
   d) Run the following command to catalog the remote node:

```
setup -t catalogRemoteDB2Nodes
```

What to do next

Redeploy i2 Analyze so that the configuration changes are deployed to the application. For more information about redeploying your system, see “Deploying i2 Analyze” on page 150.

Changing the database management system

If you want to use a different database management system in your deployment, you need to update the details in the topology and, if appropriate, provide an alternative JDBC driver.

About this task

Important: The Information Store can be deployed on only DB2.

The database management system that is installed into your environment needs to be identified within the toolkit to ensure that the database interactions are correctly tailored. In addition, the toolkit needs to contain the correct JDBC driver for the installed database management system. To change the database management system, check that the details match the values that are specified in the toolkit.

Procedure

1. To change the database management system, copy the version of `topology.xml` that is specific to the database management system you are using from the `examples` directory into `toolkit\configuration\environment`.

   The topology files are in the following location: `toolkit\configuration\examples\topology\`*`database management system type`*`\topology.xml`.

   Note: If you are using Oracle, you must populate the `instance-name` attribute.
2. On the i2 Analyze server, open a command prompt and navigate to `toolkit\scripts`.

3. To set the default values, run the following command:

```
setup -t generateDefaults
```

4. Check the default values in each file to ensure that they match your environment, and if required modify the values.

What to do next

After you change the database management system, you must change the JDBC driver to match. For more information about changing the driver, see “Changing the JDBC driver” on page 17.

Changing the JDBC driver

The application server requires a JDBC driver to enable communication with the database. The JDBC driver that you provide to the deployment toolkit depends on the database management system to use with the deployment.

About this task

You must ensure that the correct JDBC driver is in use. Depending on the type of database management systems that are installed, the JDBC drivers that you provide differ. Choose the JDBC drivers that relate to the database management systems that are installed.

Procedure

1. Locate the JDBC driver for your database management system.
   - If you are using DB2, locate the `IBM\SQLLIB\java\db2jcc4.jar` file.
   - If you are using SQL Server, download the Microsoft JDBC Driver 6.0 for SQL Server from https://www.microsoft.com/en-us/download/confirmation.aspx?id=11774. Extract the contents of the download, and locate the `sqljdbc_6.0\enu\jre8\sqljdbc42.jar` file.
   - If you are using Oracle, locate the `Product\11.2.0\dbhome_1\jdbc\lib\ojdbc6.jar` file on Windows, or the `/app/oracle/product/11.2.0/dbhome_1/jdbc/lib/ojdbc6.jar` file on Linux.
2. Copy the relevant driver file for your database management system to the `toolkit\configuration\environment\common\jdbc-drivers` directory.
3. Optional: To use a JDBC driver `.jar` file that has a custom name, you must add the following attribute to the `<database>` element in your `topology.xml` file: `jdbc-driver="`*`custom_name`*`.jar"`.

   Where *custom_name* is the name of the `.jar` file in the `toolkit\configuration\environment\common\jdbc-drivers` directory.

What to do next

After you update each property file, you can either modify other aspects of the deployment toolkit or redeploy the system to update the deployment with any changes. For more information about redeploying your system, see “Deploying i2 Analyze” on page 150.

Modifying the maximum number of notes in a record

In a deployment that contains the Opal services, analysts can add notes that contain information that is not categorized by the type of entity or link to Information Store records. The number of notes that a record contains can impact the performance of searching for and uploading that record.

About this task

You can configure the maximum number of notes that a record can contain to a number that better matches system requirements. By default, the maximum number of notes that a record can contain is 50.

If a record already contains more notes than the maximum that you set, analysts can continue to edit and delete the existing notes. However, analysts can only create notes in the record by deleting existing notes until the number of notes is less than the new maximum.

Procedure

1. Using a text editor, open the `DiscoServerSettingsCommon.properties` file. You can find this file in the following location: `toolkit\configuration\fragments\opal-services-is\WEB-INF\classes`.
2. Change the value of the `RecordMaxNotes` property.
   For example, `RecordMaxNotes=25`.
3. Save your changes.

What to do next

After you modify the maximum number of notes a record can contain, you must redeploy i2 Analyze for the changes to take effect. For more information about redeploying your system, see "Redeploying i2 Analyze" on page 98.

After you redeploy i2 Analyze, test that the system continues to meet your requirements.

Specifying the connection URI

If your deployment contains the Opal services and uses an extra proxy server between the clients and i2 Analyze, you must specify the URI that clients use to connect to i2 Analyze. Clients that do not use the specified URI are unable to connect to i2 Analyze.

Before you begin

Your proxy server must be configured so that any security configurations, for example Secure Sockets Layer or client certificate authentication, are passed through to the i2 Analyze server. You must complete this according to the documentation for your proxy server.

About this task

The example deployments use a single HTTP server proxy on the i2 Analyze server that clients connect to. Your system might include an extra proxy server between the clients and the i2 Analyze server. Usually this extra proxy server has a different URI to i2 Analyze. If users try to connect to i2 Analyze using the URI of the proxy server, they are unable to connect to i2 Analyze. To allow users to connect by using the URI of the proxy server, you must specify that URI in the i2 Analyze configuration. After you configure the connection URI for your deployment, users that connect to i2 Analyze must use the URI that you specify.

Note: You can specify only one URI that clients can use to connect to your deployment of i2 Analyze.

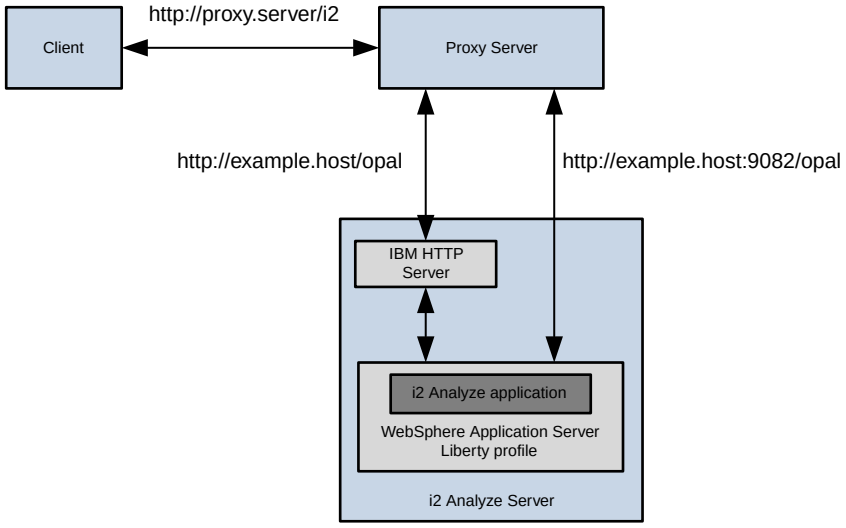
When clients connect to your proxy server using the connection URI that you specify, you must configure your proxy server to route requests from the client to the URI of the i2 Analyze server. The

URI of the i2 Analyze server is defined in the topology file. The URI is determined from the `host-name` attribute of the `<application>` element, and the `context-root` attribute of the `<war>` element. For example, the URI for a deployment with the following topology file is `http://example.host/opal`.

```
<application http-server-host="true" name="opal-server" host-name="example.host">
  <wars>
    <war context-root="opal" name="opal-services-is" i2-data-source-id="infostore"
     target="opal-services-is">
  ...
</application>
```

Note: If your Opal application has the `http-server-host` attribute set to `false`, your proxy server must route requests with the port number of the application. You can find the port number of the application in the `toolkit\configuration\environment\opal-server\port-def.props` file.

The following image shows the URIs that you might use in this example:



Procedure

1. Using a text editor, open the `toolkit\configuration\fragments\opal-services-is\WEB-INF\classes\DiscoClientSettings.properties` file.
2. Set the value of the `FrontEndURI` property to the URI that can be used to connect to your deployment of i2 Analyze through the proxy server.
   For example, `FrontEndURI=http://proxy.server/i2`.
3. Save and close the file.

What to do next

After you redeploy i2 Analyze and start the server, the URI that can be used to access the system is displayed in the console. Ensure that you can access i2 Analyze by using this URI from a client workstation that uses the proxy server.

Modifying the credentials

To allow the deployment toolkit to update the database, Lightweight Third-Party Authentication (LTPA) keys, and Solr search platform components, you must provide user names and passwords. The user names and passwords that you provide allow the system to set up and administer components of i2 Analyze, and are not used to access i2 Analyze.

About this task

Three types of credentials are stored in `credentials.properties`:

- Database
- LTPA keys
- Solr search platform

Note: Solr search platform credentials are only required for a deployment that uses the Information Store.

You might need to change the values in the `credentials.properties` file from the values that are used in the example deployment. For example, IBM DB2 might require a different user name and password in a test or production system.

When you deploy i2 Analyze, the passwords in the `credentials.properties` file are encoded.

Database

For each database that is identified in `topology.xml`, you must specify a user name and a password in the `credentials.properties` file. The `setup` script uses this information to authenticate with the database.

Note: The user that you specify must have privileges to create and populate databases in the database management system.

The database credentials are stored in the following format:

```
db.identifier.user-name=user name
db.identifier.password=password
```

The name of each database credential has three parts, two of which are variable:

Part	Description
`identifier`	The database identifier, for example `infostore`.
`user-name` or `password`	Indicates whether the value is a user name or a password.

For example:

```
db.infostore.user-name=admin
db.infostore.password=password
```

Note: The `db.identifier.truststore.password` credential is used only when you configure the connection between the database and Liberty to use SSL. If you are not using SSL to secure this connection, you do not need to specify a value for the `db.identifier.truststore.password` credential. For more information about configuring SSL, see Configure Secure Sockets Layer with i2 Analyze.

LTPA keys

You must provide a value for the `ltpakeys.password` property. This value is used by the system to encrypt LTPA tokens.

- For a stand-alone deployment of i2 Analyze, you can specify any value as the password.
- For a deployment of i2 Analyze that uses LTPA tokens to authenticate with other systems, you must specify the same password that those systems use.

Solr search platform

The Solr search platform is used to search data in the Information Store. You must provide values for the `solr.user-name` and `solr.password` properties. The Solr index is created when i2 Analyze is deployed, and the values that you provide here become the Solr user name and password.

Procedure

1. Using a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
2. Modify any database or LTPA credentials that need to be updated:
   - the database user names and passwords
   - the LTPA keys password
3. Leaving the current Solr properties unchanged, add any changes to the Solr credentials as new properties.

   The new properties must be created in the following format:

   ```
   solr.user-name.new=value
   solr.password.new=value
   ```

   For example:

   ```
   #Solr credentials
   #The user name and password for solr to use once deployed
   solr.user-name=admin
   solr.password={enc}E3FGHjYUI2A\=

   solr.user-name.new=admin1
   solr.password.new=password
   ```

   As a part of the deployment process:

   - The new values will replace the original values in the file
   - The password value will be encrypted
   - Once changed, the `credentials.properties` file will be cleaned, removing the duplicate values
4. Save and close the `credentials.properties` file.
5. Redeploy the system. In a command prompt, navigate to the `toolkit\scripts` directory and run the following command:

   ```
   setup -t deploy
   ```

Adding more data sources

Depending on the original example that was deployed you might wish to extend your system to use different sources of data. Extending your system in this manner, not only allows you to gain access to different data, but also to different feature sets.

Note: If you have a system that includes multiple data sources, generally these have been set up to use different server profiles. There are multiple tasks that are able to function in an environment with a single profile that need to identify the correct profile in situations where there are multiple. To specify the server profile, you must add the `-s` flag with the name. For example, to start the server profile for the Information Store, run the following command:

```
setup -t start -s opal-server
```

To start the server with the Analysis Repository, run the following command:

```
setup -t start -s onyx-server
```

Adding an Information Store

You can add an Information Store to a deployment of i2 Analyze that uses the Analysis Repository. In a deployment with both data stores, data in both stores can be accessed using Analyst's Notebook Premium, and data in the Analysis Repository can also be accessed using the Intelligence Portal.

Before you begin

Before you add an Information Store to a deployment that uses the Analysis Repository, you must ensure that your security schema does not use `READ_CLOAKED` as a dimension level value.

About this task

Only one Information Store instance can be added per deployment, but this instance can be added to an existing deployment that is currently configured to only use the Analysis Repository. This task covers creating an Information Store configured to use the Opal pattern of deployment.

Procedure

1. Run the `addInformationStore` task to add the Information Store configuration properties. In a command prompt, navigate to the `toolkit\scripts` directory and run the following command:

   ```
   setup -t addInformationStore
   ```

2. Add required credentials to the `credentials.properties` file.

   a) Using a text editor, open the `toolkit\configuration\environment\credentials.properties` file.

   b) Add the following database and Solr credentials to the file:

   ```
   db.infostore.user-name=
   db.infostore.password=

   solr.user-name=
   solr.password=
   ```

   c) Specify a user name and password to use with the Information Store database.

Note: The user that you specify must have privileges to create and populate databases in the database management system.

d) Specify a user name and password to use with Solr.

The Solr index is created when i2 Analyze is deployed, the values that you provide here become the Solr user name and password.

3. Deploy the example Information Store. In a command prompt, navigate to the `toolkit\scripts` directory and run the following command:

```
setup -s opal-server -t deployExample
```

4. To populate your Information Store with the provided example data, run the following command:

```
setup -s opal-server -t ingestExampleData
```

Adding an Analysis Repository

You can add an Analysis Repository to a deployment of i2 Analyze that uses the Information Store deployed using the Opal pattern. In a deployment with both data stores, data in both stores can be accessed using Analyst's Notebook Premium, and data in the Analysis Repository can also be accessed using the Intelligence Portal.

About this task

Only one Analysis Repository instance can be added per deployment, but this instance can be added to an existing deployment that is currently configured to only use the Information Store. This task covers creating an Analysis Repository configured to use the Onyx pattern of deployment.

Procedure

1. Using an XML editor, open `toolkit\configuration\environment\topology.xml`.
2. Add the following elements:

a) In the `i2-data-sources` section add the `ar-id`:

```
<i2-data-source id="ar-id" ar="true" default="true">
    <!-- DataSource Id is deliberately blank and should be left so -->
    <DataSource Id="" Version="0">
        <Shape>Repository</Shape>
        <Name>Analysis Repository</Name>
    </DataSource>
</i2-data-source>
```

b) In the `databases` section add the `write1` database:

```
<database dialect="db2" xa="false" instance-name=""
    database-name="WriteSto" database-type="WriteStore" id="write1"
    host-name="" port-number="50000" />
```

c) In the `applications` section add the `onyx-server` application:

```
<application name="onyx-server" host-name="" http-server-host="true">
    <lucene-indexes>
        <lucene-index id="ar" main-index-location=""
            alternatives-location="" />
    </lucene-indexes>
    <wars>
        <war target="onyx-services-ar" name="onyx-services-ar"
               i2-data-source-id="ar-id"
              context-root="apollo">
            <data-sources>
                <data-source database-id="write1" create-database="true" />
            </data-sources>
            <lucene-index-ids>
                <lucene-index-id value="ar" />
            </lucene-index-ids>
            <fragments>
                <fragment name="common" />
                <fragment name="onyx-services-ar" />
            </fragments>
        </war>
    </wars>
</application>
```

3. Add required credentials to the `credentials.properties` file.
   a) Using a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
   b) Add the following database credentials to the file:

```
db.write1.user-name=
db.write1.password=
```

   c) Specify a user name and password to use with the Analysis Repository database.

   Note: The user that you specify must have privileges to create and populate databases in the database management system.
4. Deploy the example Analysis Repository. In a command prompt, navigate to the `toolkit\scripts` directory and run the following command:

```
setup -s onyx-server -t deployExample
```

Connecting to external data sources

In addition to the Information Store, you can add data source connectors to a deployment of i2 Analyze that uses the Analysis Repository. This allows you to search for data available in these sources and to copy that information into the Analysis Repository as a record of your investigation.

There are three ways to deploy a data connector:

iBase Connector

If the external data source is iBase, then you can use the deployment toolkit and the i2 Analyze documentation to deploy and configure an iBase connector. In the Intelligence Portal, users can

perform many of the same operations on the iBase database that they can perform on the Analysis Repository. For more information about connecting to an iBase repository, see Deploying an iBase Connector.

Connector Creator

If the external data source is a relational database, then you can deploy and configure Connector Creator instead of writing code in Developer Essentials. In this approach, Connector Creator presents its own user interface in the Intelligence Portal, and users can choose from a preconfigured set of parameterized searches to run. For more information about creating a connection to relational database, see "Deploying Connector Creator" on page 36.

Other connections

For other external data sources, you can use i2 Analyze Developer Essentials to create and deploy a custom data connector. The operations that users can perform then depend on the implementation of the connector. For more information about connecting to other data sources, see IBM i2 Analyze Developer Essentials.

Deploying an iBase Connector

As part of an IBM i2 Analyze deployment, an iBase connector provides a mechanism for providing users with access to an IBM i2 iBase database. The database becomes available in the Intelligence Portal as a data source that users can select and interact with.

There are two reasons for integrating iBase with a deployment of i2 Analyze. One reason is that you already have an iBase deployment, and you want to retain your data and your data model as you upgrade or migrate to i2 Analyze. The other reason is that you already have a deployment of i2 Analyze 4.1, and you want to use iBase as a way to ingest data from other data sources.

The deployment procedure that you follow for an iBase connector depends on your starting point. However, when the procedure is complete the combined deployment has the following features:

- Users can browse the entities, links, images, and documents that are stored in iBase.
- When a user runs a free text search through the Intelligence Portal, the connector uses iBase Search 360 to search for exact matches and variants.
- When a user performs a property search for entities or links, the connector maps it to an equivalent iBase query.
- When a user creates a visual query in the Intelligence Portal, the connector translates it into an iBase query.

  Note: iBase connectors do not support the use of count conditions to search for entities that are linked to a specific number of other entities.
- When a user clicks **Show context** to view the entities and links that are related to a search result, the connector performs the operation against the data in iBase.

About this guide

The instructions describe how to deploy an iBase connector in i2 Analyze. For more information about prerequisites and supported environments, see the Release Material.

Note: This version of i2 Analyze supports one iBase connector per deployment.

The instructions assume that you have access to a distribution or a deployment of iBase 8.9.11 that is running on a compatible version of Microsoft SQL Server. The deployment process for the connector also requires a Java™ Database Connectivity (JDBC) driver that you can download from the Internet when you need it.

The instructions assume that you already have a working deployment of i2 Analyze with the Onyx services. If you are deploying an iBase connector in a solution that uses both the Onyx and Opal services, you must specify the Onyx server profile when you run any of the setup commands. For example, `setup -t addIBaseConnector -s onyx-server`.

The instructions for an iBase connector are detailed in the following sections. The essential steps are summarized in the list.

1. Synchronize the iBase and i2 Analyze schemas.
2. Configure the iBase connector.
3. Deploy the iBase connector.

Synchronizing iBase and i2 Analyze schemas
Before you can configure and deploy an iBase connector, the deployments of i2 Analyze and iBase that it connects must have compatible data structures. Depending on your starting point, you must deploy i2 Analyze with a schema that matches an iBase database, or iBase with a schema that matches a deployment of i2 Analyze.

About this task

i2 Analyze includes tools for creating an iBase schema from an i2 Analyze schema, and for performing the opposite operation. To use an iBase connector, you need working deployments of iBase and i2 Analyze whose schemas match.

If you want to set up a new i2 Analyze deployment that uses an existing iBase database, you must create an i2 Analyze schema from that database. Then, you use this schema when you deploy i2 Analyze. Follow the instructions in “Deploying i2 Analyze with a schema derived from iBase” on page 26.

If you want to create an iBase database to use as a warehouse for an existing i2 Analyze deployment, you must generate the iBase schema from i2 Analyze. Then, you use this schema when you deploy iBase. Follow the instructions in “Deploying iBase with a schema derived from i2 Analyze” on page 29.

Deploying i2 Analyze with a schema derived from iBase
To make an existing iBase database available through a new deployment of i2 Analyze, you must first generate an i2 Analyze schema that is compatible with the iBase database. Then, deploy i2 Analyze. Finally, follow the instructions to configure and deploy the iBase connector.

Before you begin

Some iBase databases require modification or administration so that you can access them through an iBase connector. There are checks that you must perform before you do anything else:

- Ensure that your iBase deployment is at version 8.9.11, and that it is running on a compatible version of Microsoft SQL Server, and that Search 360 is enabled.
- Use the Valid End Types tool in iBase Designer to ensure that the data in the database is properly aligned with its schema.
- In the network configuration for SQL Server, ensure that the TCP/IP protocol is enabled.
- Ensure that the SQL Server and SQL Server Browser services are both running on your server.
- Ensure that you have access to an SQL Server account that is a member of both the `db_datareader` and `db_datawriter` roles on the iBase database.

About this task

Mostly, the procedure for deploying i2 Analyze with a schema that is derived from iBase is the same as the procedure for any other i2 Analyze deployment. The main differences come at the start, when you use the deployment toolkit and the Analysis Repository Tools to create and then refine the schema to use.

Procedure

1. Install, but do not deploy, i2 Analyze according to the instructions in the Installing IBM i2 Analyze.
2. Create the configuration directory:
   a) Navigate to the `\toolkit\examples\configurations\analysis-repository` directory.
   b) Copy the configuration directory to the root of the toolkit.
3. Download the `jtds-1.2.4-dist.zip` file from https://sourceforge.net/projects/jtds/files/jtds/1.2.4 and extract its contents.
   This file contains a JDBC driver for Microsoft SQL Server.
4. Copy the extracted `jtds-1.2.4.jar` file to the `toolkit\configuration\environment\common\jdbc-drivers` directory of your i2 Analyze installation.
5. Use a text editor to open the file at `toolkit\configuration\environment\iBase \environment.properties`.
6. Specify values for the following properties, and then save and close the file:

Name	Value
`i2analyze.schema`	The name of the generated i2 Analyze schema file, which is saved to the `toolkit\configuration\fragments\common\WEB-INF\classes` directory. When you generate the schema, a properties file named `iBaseToi2AnalyzeMappings.properties`, which maps iBase names to i2 Analyze type names, is saved to the same directory.
`ibase.hostname`	The host name of the server where the iBase database is deployed.
`ibase.database`	The name in SQL Server of the iBase database from which to create the i2 Analyze schema.
`ibase.username`	The user name of an SQL Server account that has permission to insert and update iBase data.
`ibase.password`	The password for the SQL Server account that has permission to insert and update iBase data.
`ibase.instance`	The name of the SQL Server instance that hosts the iBase database.
`path.to.jtds.jar`	The full path to the directory that contains the jTDS driver file. **Tip:** If your path contains spaces, you must enclose the value within double quotation marks.
`jtds.jar.name`	The name of the jTDS driver file, which is likely to be `jtds-1.2.4.jar`.

These property values are the only ones that you need to specify for this part of the deployment process. You provide values for the other properties in `environment.properties` later on.

7. Provide your new deployment with some basic configuration settings:
   a) On the i2 Analyze server, open a command prompt and navigate to `toolkit\scripts`.
   b) To set the default values, run the following command:

```
setup -t generateDefaults
```

   c) Check the default values in each file to ensure that they match your environment, and modify them if required.
8. At the command prompt that you opened in the previous step, run the following command:

```
setup -t createI2AnalyzeSchemaFromIBase
```

   The i2 Analyze schema file that you specified in the `i2analyze.schema` property, and its associated `iBaseToi2AnalyzeMappings.properties` file, are generated in the `toolkit\configuration\fragments\common\WEB-INF\classes` directory.

   Important: For security reasons, running the `createi2AnalyzeSchemaFromIBase` task deletes the plain-text SQL Server password from the `environment.properties` file. When you run more tasks that need access to the iBase database, you must reenter the password and save the file again.

The generated i2 Analyze schema contains definitions for entity types and link types that match the definitions in iBase. The default definitions have the following characteristics:

- Entity types and link types all use the same icon.
- Property types do not appear in a predictable order.
- Item labels are set to the value of the `DisplayName` properties on those items.

Before you deploy i2 Analyze with the generated schema, you can use IBM i2 Analyze Schema Designer to improve it.

9. Install the Schema Designer from the `tools\schema-designer` directory of the i2 Analyze distribution media.
   For more information, see the Schema Designer documentation.
10. Edit the schema to match your needs. There are several changes that you might consider:
    - Specify different default icons for different entity and link types
    - Change how label generation works.
    - Create a charting scheme for the new schema.
    - Change the order of the property types within each item type.

When you are content with the schema, you can deploy i2 Analyze.

11. Use a text editor to open `toolkit\configuration\fragments\common\WEB-INF\classes\ApolloServerSettingsMandatory.properties`.
12. Set the value of the `SchemaResource` property to match the value that you set for `i2analyze.schema` in the `environment.properties` file for the iBase connector.
13. Specify the credentials:
    a) Using a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
    b) Enter the user name and password to use with the database.
    c) Enter the password to use to encrypt LTPA tokens.

d) Save and close the `credentials.properties` file.

14. Run the setup script to create the example deployment:
    a) On the i2 Analyze server, open a command prompt and navigate to the `toolkit\scripts` directory.
    b) To deploy the example, run the following command:

```
setup -t deployExample
```

    c) To start the application server, run the following command:

```
setup -t start
```

    d) Start, or restart, the HTTP server that hosts the reverse proxy.
15. Open the Intelligence Portal, and test the deployment to ensure that it behaves correctly. If necessary, adjust the settings, redeploy, and retest.

Results

You now have a deployment of i2 Analyze with a schema that is compatible with an existing iBase database. Everything else about the deployment of i2 Analyze has its default or example behavior.

What to do next

To enable users to access iBase through the Intelligence Portal, configure and deploy an iBase connector on the i2 Analyze server.

Deploying iBase with a schema derived from i2 Analyze

To add a new iBase database to an existing i2 Analyze deployment, you must first generate a database schema from the i2 Analyze schema. Then, apply the schema to a new iBase database and populate the database. Finally, follow the instructions to configure and deploy the iBase connector.

Before you begin

Some deployments of i2 Analyze require modification or administration so that you can augment them with an iBase connector. You must perform these checks before you do anything else:

- Ensure that your i2 Analyze deployment is at a supported version (4.1.1. or later).
- If your i2 Analyze deployment does not already use an SQL Server database, download `jtds-1.2.4-dist.zip` from https://sourceforge.net/projects/jtds/files/jtds/1.2.4 and extract its contents.
- Copy the extracted `jtds-1.2.4.jar` file to the `toolkit\configuration\environment\common\jdbc-drivers` directory of your i2 Analyze installation.

About this task

The procedure for creating an iBase database with a schema that reflects the entity, link, and property types in an i2 Analyze deployment starts with a new, empty iBase database. The next steps are to configure i2 Analyze with information about iBase, and then to generate, apply, and improve the new schema.

Procedure

1. Install iBase 8.9.11 and a compatible version of Microsoft SQL Server according to the instructions in the iBase Release Notes.
2. Use iBase Designer to create an empty iBase database in your SQL Server instance.

3. Create an SQL Server account that is a member of the `db_datareader` and `db_datawriter` roles on the new iBase database.
4. On the i2 Analyze server, use a text editor to open the file at `toolkit\configuration\environment\iBase\environment.properties`.
5. Specify values for the following properties:

Name	Value
`i2analyze.schema`	The file name of the i2 Analyze schema in the `toolkit\configuration\fragments\common\WEB-INF\classes` directory of the existing i2 Analyze deployment.
`ibase.hostname`	The host name of the server where the iBase database is deployed
`ibase.database`	The name in SQL Server of the empty iBase database that you created
`ibase.username`	The user name of an SQL Server account that has permission to insert and update iBase data
`ibase.password`	The password for the SQL Server account that has permission to insert and update iBase data
`ibase.instance`	The name of the SQL Server instance that hosts the iBase database
`path.to.jtds.jar`	The full path to the directory that contains the jTDS driver file **Tip:** If your path contains spaces, you must enclose the value within double quotation marks.
`jtds.jar.name`	The name of the jTDS driver file, which is likely to be `jtds-1.2.4.jar`

   These property values are the only ones that you need to specify for this part of the deployment process. You provide values for the other properties in `environment.properties` later on.
6. On the i2 Analyze server, open a command prompt and navigate to the `scripts` directory of the i2 Analyze deployment toolkit.
7. Run the following command:

```
setup -t replaceIBaseSchemaFromI2Analyze
```

   The command populates the metadata tables in the iBase database, but does not create the database tables and indexes.

   Important: For security reasons, running the `replaceIBaseSchemaFromI2Analyze` task deletes the plain-text SQL Server password from the `environment.properties` file. When you run more tasks that need access to the iBase database, you must reenter the password and save the file again.
8. In iBase Designer, run the Schema Integrity Check tool twice consecutively on the iBase database to create all the necessary objects.

   For more information about the Schema Integrity Check tool, see the iBase Designer help.
9. Optional: Use iBase Designer to make improvements to the iBase schema that do not affect the structure of the data.
   For example, change the representations of iBase item types, or assign semantic types to them.

10. Enable Search 360 on the iBase database.

    For more information, see the iBase Administration Center.

Results

You now have an iBase database with a schema that is compatible with an existing deployment of i2 Analyze. Everything else about the iBase database has its default behavior.

What to do next

After you create the iBase database, you can populate it with your data. After you populate the database, you can configure and deploy an iBase connector on the i2 Analyze server to enable users to access iBase through the Intelligence Portal.

Configuring and deploying an iBase connector

When you have deployments of IBM i2 Analyze and iBase whose schemas are synchronized, you can configure and deploy an iBase connector on the i2 Analyze server. Users are then able to access iBase data through the Intelligence Portal.

Before you begin

To deploy an iBase connector successfully, you must first have completed deployments of i2 Analyze and iBase 8.9.11 whose schemas are synchronized in one of the following ways.

- Generate a new iBase database from an existing i2 Analyze schema by following the instructions in "Deploying iBase with a schema derived from i2 Analyze" on page 29, set the value to `1`.
- Create an i2 Analyze schema from an existing iBase database by following the instructions in "Deploying i2 Analyze with a schema derived from iBase" on page 26, set the value to `2`.

In addition, Search 360 must be enabled on the iBase database.

About this task

The process of configuring and deploying an iBase connector involves the following steps:

- Provide values for the remaining properties in the `environment.properties` file
- Enable i2 Analyze to access binary documents in the iBase database
- Use the deployment toolkit to add an iBase connector to your i2 Analyze configuration
- Redeploy i2 Analyze with the iBase connector

If you use auditing in iBase, you can also configure the connector to log the iBase queries that Intelligence Portal users submit to the audit database.

Configuring an iBase connector

When you have compatible deployments of i2 Analyze and iBase, you can configure an iBase connector to work between them. The procedure involves providing settings that affect how the connector behaves, and changing some HTTP server settings. You can also enable iBase to audit the queries that users make through the Intelligence Portal.

Procedure

1. Using a text editor, open the file at `toolkit\configuration\environment\iBase \environment.properties`.
2. Specify values for the following properties:

Name	Value
ibase.origin.db.name	A name for the connected iBase database that identifies it in some automatically generated configuration files. This property can have any value, but it must not be empty. It is conventional to set it to the same value as ibase.database.
ibase.datasource.jndi	The JNDI name for the data source, which is also used in the configuration files for an iBase connector. This property can have any value, but it must not be empty. By convention, JNDI names start with "ds/"; for example, ds/iBase.
ibase.datasource.id	The data source identifier for the iBase database, which is used in some of the iBase connector configuration files. This property can have any value, but it must not be empty. It is common for it to match ibase.datasource.jndi, but without the "ds/" prefix.

ibase.security.context	The security dimension values that items from iBase receive. You can opt to specify that items in iBase with different security classification codes (SCCs) receive different dimension values, but you *must* provide a default set with the following syntax:
	`ibase.security.context=Default,HI,OSI,CON`
	Here, Default is a fixed string, while HI, OSI, and CON are dimension value identifiers. You must ensure that you identify at least one value from each dimension in your security schema.
	Note: If you want items from iBase to receive the same security settings as items in the Analysis Repository receive by default, arrange for the setting here to be consistent with the <DefaultItemSecurityTags> setting in toolkit\configuration\fragments\core\ApolloClientSettings.xml.
	If your iBase deployment uses SCCs, then you can specify a different set of dimension values for each SCC, like this example:
	`ibase.security.context=`
	`Default,HI,OSI,CON||SCC_1,HI,CON||SCC_2,OSI,UC`
	Any item with an SCC that does not appear in the ibase.security.context property value automatically receives the Default settings.

ibase.operation.mode	An indicator of whether you began your iBase connector deployment with an existing deployment of i2 Analyze or iBase. You must give this property one of the following values: • If you generated a new iBase database from an existing i2 Analyze schema by following the instructions in "Deploying iBase with a schema derived from i2 Analyze" on page 29, set the value to 1.

Name	Value
	• If you created an i2 Analyze schema from an existing iBase database by following the instructions in "Deploying i2 Analyze with a schema derived from iBase" on page 26, set the value to `2`.
`ibase.max.result.count`	The maximum number of results that the iBase connector returns in a single operation. This property can have any value, but it must not be empty. The purpose of the property is to prevent users from requesting large data sets from the database. The count is based only on the number of matches returned in response to a query, it does not measure of the amount of data in each match instance. The appropriate setting for your deployment depends on your deployment pattern, your data, and the type of results required. Start with an average value, for example 200, revise upwards if you need to see more results or downwards if you are looking for improved performance. **Note:** Note: In the Onyx pattern, the maximum number of results that can be selected and added to the chart is 500.
`ibase.library.id`	The identifier for the jTDS driver file, which is used in the configuration files for an iBase connector. This property can have any value, but it must not be empty. For example, `jTDSLib`.
`ibase.audit.required`	If you use auditing in iBase, you can configure the connector to log all user queries to the iBase audit database. This property must be set to one of the following values: • If you did not configure auditing in iBase, or you do not want the connector to log user queries, set the value to `false`. • If you configured auditing and created an audit database in iBase, and you want to log all the queries that are submitted to iBase through the Intelligence Portal, set the value to `true`.

3. If you set `ibase.audit.required` to `true`, then you must specify values for three more properties:

Name	Value
`ibase.audit.database.name`	The name in SQL Server of the iBase audit database.
`ibase.audit.jndi`	The JNDI name for the data source that represents the iBase audit database. This property can have any value, but it must not be empty when `ibase.audit.required` is `true`. By convention, JNDI names start with `"ds/"`; for example, `ds/audit`.
`ibase.datasource.audit.id`	The data source identifier for the iBase audit database. This property can have any value, but it must not be empty when `ibase.audit.required` is `true`. It is common for this property to match `ibase.datasource.jndi` without the `"ds/"` prefix.

You must also ensure that the SQL Server account that you specified in `ibase.username` is a member of the `db_datareader` and `db_datawriter` roles on the iBase audit database.

The final part of configuration is to enable i2 Analyze to access binary documents in the iBase database by adding a rewrite condition and a rewrite rule to the HTTP server.

4. Using a text editor, open the `http-custom-rewrite-rules.txt` file from the `configuration\environment\proxy` directory of the i2 Analyze deployment toolkit.
5. Add the following code to the file, between the lines that contain `!Start_After_Rules!` and `!End_After_Rules!`:

```
RewriteCond %{QUERY_STRING} ^documentId=DA-servlet/([^&]+)
RewriteRule ^/${contextRoot:onyx-services-ar}/services/download
            /${contextRoot:iBase}/services/dadownload?documentId=%1 [PT]
```

   Ensure that you enter the rewrite rule on a single line, omitting the line break in the example, but including a space after `download`.
6. Save and close the text file.

   You are now ready to install and deploy your iBase connector.

Installing and deploying an iBase connector

When you have a set of configuration files that meet your requirements, you can deploy the iBase connector. The following procedure describes how to add the connector and information about the iBase data source to your existing i2 Analyze deployment.

Procedure

1. On the i2 Analyze server, open a command prompt and navigate to the `scripts` directory of the i2 Analyze deployment toolkit.
2. Run the following commands:

```
setup -t addIBaseConnector
setup -t addIBaseDatasource
```

   Important: For security reasons, running the `addIBaseDatasource` task deletes the plain-text SQL Server password from the `environment.properties` file. When you run more tasks that need access to the iBase database, you must reenter the password and save the file again.
3. Use the following command to stop the application nopserver:

```
setup -t stop
```

4. Redeploy i2 Analyze to add the iBase connector to the deployment, and to update IBM HTTP Server:

```
setup -t deploy
```

5. Use the following command to start the application server:

```
setup -t start
```

6. Restart IBM HTTP Server.

Updating an iBase connector
If you need to modify the behavior of an iBase connector, the procedure for updating it is similar to the procedure for the original deployment.

Procedure

1. On the i2 Analyze server, use a text editor to open the file at `toolkit\configuration\environment\iBase\environment.properties`.
2. Make your changes to the configuration.
   For example, you might enable auditing, or reduce the maximum number of results from an iBase search.
3. Reenter the SQL Server account password, and then save and close the file.
4. Open a command prompt and navigate to the `scripts` directory of the i2 Analyze deployment toolkit.
5. Run the following sequence of commands:

```
setup -t addIBaseConnector
setup -t addIBaseDatasource
setup -t stop
setup -t deploy
setup -t start
```

   Note: You reuse the `addIBaseConnector` and `addIBaseDatasource` tasks, even though you are updating rather than adding these components.
6. Restart IBM HTTP Server.

Results
After you complete this procedure, the i2 Analyze deployment is running again, with your changes to the iBase connector in place.

Configuring custom logging for an iBase connector
By default, an iBase connector inherits its logging settings from its parent i2 Analyze deployment. You can change this behavior so that iBase connector logging uses a different level, or is saved in a different location.

About this task

By default, the logging level for i2 Analyze is defined in a file named `log4j.properties`, in the `configuration\fragments\common\WEB-INF\classes` directory of the i2 Analyze deployment toolkit.

In an unmodified deployment, i2 Analyze sends the messages from a running iBase Connector to a file named `IBM_i2_Analysis_Repository.log`, in the `usr\servers\onyx-server\logs\onyx-services-ar` directory of WebSphere Application Server Liberty profile. The logging level is `WARN`, which logs potentially harmful situations, error events, and unrecoverable error events, but not informational messages.

Procedure

To configure different logging specifically for an iBase connector, follow these instructions:

1. Create a separate `log4j.properties` file in the `configuration\fragments\iBase\WEB-INF\classes` directory.
2. Populate the new `log4j.properties` file to reflect your requirements.

You can describe how to handle log messages from the following iBase connector libraries:

- `com.i2group.connector.common`
- `com.i2group.connector.common.adapter`
- `com.i2group.connector.common.data`
- `com.i2group.connector.common.data.basic`
- `com.i2group.connector.common.editor`
- `com.i2group.connector.daod.ibase`

For more information about working with log4j, see the Apache Logging Services Project website at http://logging.apache.org.

3. Update i2 Analyze by running `setup -t deploy` from a command prompt, as usual.

Deploying Connector Creator

As part of an IBM i2 Analyze deployment, Connector Creator provides a way to access some types of external data source without the need to write a custom data connector. Following a successful deployment, Connector Creator makes data from external sources available for users to query and view in the Intelligence Portal.

About this guide

The instructions in this deployment guide describe how to add Connector Creator to a deployment of i2 Analyze that uses IBM DB2®. For more information about prerequisites and supported environments for the rest of the system, see the Release Material.

The instructions assume that you already have a working deployment of i2 Analyze with the Onyx services, and that you have access to the external databases that Connector Creator connects this deployment to.

If you are deploying Connector Creator in a deployment that uses both the Onyx and Opal services, you must specify the Onyx server profile when you run all the setup commands in this section. For example, `setup -t addConnectorCreator -s onyx-server`.

Important: For a successful deployment of Connector Creator, the Analysis Repository must be a DB2 database. The configuration mechanism does not support any other database management system.

Installing Connector Creator

The first stage of enabling access to an external data source through Connector Creator is to add Connector Creator itself to your deployment of i2 Analyze. The process creates storage for the configuration file, and adds new elements to the Intelligence Portal user interface.

Before you begin

Ensure that i2 Analyze is deployed with IBM DB2.

Procedure

1. On the i2 Analyze server, open a command prompt and navigate to the `scripts` directory of the i2 Analyze deployment toolkit.
2. Run the following command to add Connector Creator to your i2 Analyze deployment configuration:

```
setup -t addConnectorCreator
```

The command adds a fragment for Connector Creator to the `toolkit\configuration\fragments` directory, and information about the configuration database to the topology file.

3. Using a text editor, open the `ApolloServerSettingsDaodMandatory.properties` file from the new `toolkit\configuration\fragments\Connected\WEB-INF\classes` directory.
4. Delete the following line, and then save and close the file:

```
MainSearchIndex=C:/IBM/iap/daod5-data
```

5. Using an XML editor, open the `topology.xml` file from the `toolkit\configuration\environment` directory.
6. Complete the information about the configuration database by providing the `host-name` and the `instance-name`. In most circumstances, these details are the same as for the i2 Analyze data store:

```
<database database-type="ccConfig" dialect="db2" instance-name="InstanceName"
    database-name="CONFIG" xa="true" id="config"
    host-name="HostName" port-number="50000" />
```

7. Using a text editor, open the `credentials.properties` file from the `toolkit\configuration\environment` directory.
8. Provide `db.config.user-name` and `db.config.password` with the user name and password of a user with permission to add and modify the configuration database in DB2. Again, in most circumstances, these will match the credentials that you use for the i2 Analyze data store.
9. At the command prompt, run the following sequence of commands to deploy Connector Creator and restart i2 Analyze:

```
setup -t stop
setup -t deploy
setup -t start
```

10. Restart the HTTP server, and then open the Intelligence Portal in your web browser.
The application toolbar contains a new **Connected** button that enables users to work with the external data sources that are connected through Connector Creator. However, no data sources are available until you also add those to the i2 Analyze deployment.

What to do next

After you install Connector Creator, the next part of the deployment process is to add information about the external data sources that users will run queries against.

Adding an external data source for Connector Creator

The second stage of enabling access to an external data source through Connector Creator is to provide the i2 Analyze deployment with information about that source. Depending on the database engine of the external source, you might also need to provide an additional JDBC driver.

About this task

Connector Creator uses the support in WebSphere Liberty Profile for connecting to relational databases. For more information about connecting to different database engines, see the topic named Configuring database connectivity in Liberty, in the Liberty documentation.

Procedure

1. Obtain a Java Database Connectivity (JDBC) driver for the external database that you want to connect to.

   Note: If the external database is hosted on DB2, then you can use same JDBC driver that the i2 Analyze data store and the Connector Creator configuration database use.
2. If necessary, save the driver file to the `configuration\environment\common\jdbc-drivers` directory of the i2 Analyze deployment toolkit.
3. Using an XML editor, open the `server.datasources.xml` file from the `deploy\wlp\usr\servers\onyx-server` directory of your i2 Analyze installation.
4. At the end of the file, immediately above the `</server>` closing tag, add a `<library>` element and a `<dataSource>` element for the external database to which you want to connect.
   For example, the following pair of elements configures access to a locally hosted SQL Server database named `Example`, through the jTDS JDBC driver:

   ```
   <library id="jTDSLib">
       <fileset dir="C:/IBM/i2analyze/toolkit/
                                configuration/environment/common/jdbc-drivers"
                includes="jtds-1.2.4.jar"/>
   </library>
   <dataSource id="Example" type="javax.sql.XADataSource" jndiName="jdbc/Example">
       <jdbcDriver libraryRef="jTDSLib"
                javax.sql.XADataSource="net.sourceforge.jtds.jdbcx.JtdsDataSource"/>
       <properties instance="SQLEXPRESS"
                databaseName="Example" serverName="localhost"/>
   </dataSource>
   ```

   WebSphere Liberty Profile supports connections to a range of different database engines. The precise requirements on the contents of the `<library>` and `<dataSource>` elements vary according to the database in question. For more information, see the Liberty documentation topic named Configuring database connectivity in Liberty.
5. Add `user` and `password` attributes to the `<properties>` element to specify the credentials of a user with appropriate access to the database.

   You can specify the user name in plain text, but for improved security you should encode the password:

   a) Open a command prompt and navigate to the `deploy\wlp\bin` directory of your i2 Analyze installation.

   b) Run the following command to generate an encoded version of the password:

   ```
   securityUtility encode --encoding=aes Password
   ```

   c) Copy the output of the command to the value of the new `password` attribute.
6. At the open command prompt, navigate to the `scripts` directory of the i2 Analyze deployment toolkit.
7. Run the following commands to add the external data source to your i2 Analyze deployment configuration:

   ```
   setup -t stop
   setup -t start
   ```

What to do next

After you have added information about the external data source, you must create a configuration file that identifies the data to query and maps that data to i2 Analyze entities, links, and properties.

Creating the Connector Creator configuration file

To exchange data with an external database, Connector Creator requires a configuration file that contains two kinds of information. You must define the queries that users can run, and map the results of those queries to a form that i2 Analyze can accept.

About this task

A configuration file for Connector Creator is an XML file that contains similar information for each of the external databases that Connector Creator connects to. For each connected database, the configuration file must define:

- The queries that Intelligence Portal users can run against the database, written in SQL
- Mappings from the query result structures to the property types of i2 Analyze entity and link types

The procedure here describes the process of building a Connector Creator configuration file. For more details about the contents of the file, see The Connector Creator configuration file.

Procedure

To be able to translate from query results to i2 Analyze data, you need an XML representation of the i2 Analyze schema that you can examine and create mappings to.

1. On the i2 Analyze server, open a command prompt and navigate to the `scripts` directory of the i2 Analyze deployment toolkit.
2. Run the following command to generate a `.jar` file that contains the XML representation:

   ```
   setup -t generateMappingJar -x i2analyze_schema -o definition_jar
   ```

   Here, *i2analyze_schema* is the full name (including the path) of the schema file, and *definition_jar* is the full name (including the path) of the output `.jar` file.

   Attention: If *definition_jar* specifies an existing file, the `generateMappingJar` task does not overwrite it. If you run the same command twice in succession, ensure that you move or delete the output between calls.

Inside the `.jar` file, the name of the XML file that you need is `schema4.xsd`. You might find it helpful to keep that file open in an editor as you follow the rest of this procedure to create the configuration file itself.

3. Using an XML editor, create an empty XML file with a name of your choice.
4. Create the root element of the configuration file, which must be named `<configuration>`.
5. Inside the `<configuration>` element, create one `<dataSource>` child element for each external database that you added to the `server.datasources.xml` file.
6. To each `<dataSource>` element, add `<name>` and `<description>` child elements.

   The text contents of these elements appear in the Intelligence Portal, where they identify each source to users, and provide summary information about their data.
7. To each `<dataSource>` element, add a `<specification>` child element with a text value that identifies the external database.

For each data source, the contents of the <specification> element must match the value of the jndiName attribute of the corresponding <dataSource> element in the server.datasources.xml file.

For more information about the three previous steps, see "The <dataSource> element" on page 44.

8. To each <dataSource> element, add a <securityDimensionValueIds> element that specifies the security dimension values to be assigned to all items that users retrieve from this data source.

   Add one <securityDimensionValueId> to its parent for each value that you want to assign.

   Note: Items from external data sources are subject to the same rules as items from the Analysis Repository. Every item must have at least one value from each dimension in the current i2 Analyze security schema.

For more information about the previous step, see "The <securityDimensionValueIds> element" on page 44.

9. To each <dataSource> element, add a <mappings> child element and a <resources> child element.

   Populating these two elements for each data source is the most important, and the most complicated, part of creating the configuration file.

For any data source, populating the <mappings> and <resources> elements is a recursive process:

- Each <resource> inside the <resources> element defines a broad SQL query to execute against the database, along with a set of additional clauses through which users can constrain the results. For example, the broad query might retrieve information about people from the database, while the additional clauses might allow users to search for people with particular names, or ages, or heights.
- Each <mapping> inside the <mappings> element describes how Connector Creator should map information from query results to item property values in i2 Analyze. When you have created the mappings for one resource, it is likely that you can use some of the same mappings to deal with the results from a different resource.

10. In each <resources> element, create and populate <resource> elements that define the searches that users can run against the data source.

    a) To each <resource> element, add a <specification> child element that contains the SQL query through which the data for the resource is retrieved.

    b) For each resource, identify the i2 Analyze item types that will represent the retrieved information, and then populate a <supportedMappings> element with the names of the entity and link type mappings that Connector Creator will use.

       You have not yet written them, but you can begin to determine how many mappings (that is, how many different types of information) each resource requires. Often, a resource that retrieves entities will use one mapping, while a resource that retrieves entity-link-entity structures will use three mappings.

       Important: If a resource retrieves data in which one entity can be linked to another entity of the same type, then the ends require two separate mappings even though the mapping definitions can be identical.

    For more information about this step, see "The <resources> element" on page 45.

11. For each resource in the configuration file, create at least one <query> element to define a search that users can run.

    Each query has a name and a description that appear in the Intelligence Portal, and a specification that contains the broader query from the resource specification.

    For more information about this step, see "Queries" on page 46.

12. If a query enables or requires users to provide one or more parameters, then add <parameter> elements to the query definition.

    For more information about this step, see "Parameters" on page 47.

When you have created all the resources and queries that you want to present to your users, you can write the mappings that enable them to see the results in the Intelligence Portal.

13. Determine whether the property types for the item types that you mapped are simple or complex according to your i2 Analyze XML schema definition. For the simple property types, create property mappings for each property type. For complex property types, use property groups to gather together the mappings for the component simple properties that make up the complex property type.

    For more information about this step, see “The <mappings> element” on page 49.

Results

You now have a Connector Creator configuration file that contains all of the necessary information to allow users to run a single query against an external data source. After you have successfully tested the configuration in your i2 Analyze deployment, you can return to this procedure to add further queries to Connector Creator.

What to do next

When you have a self-consistent Connector Creator configuration file, you can upload it to the configuration database on the i2 Analyze server.

Uploading and downloading the configuration file

Installing Connector Creator adds the **Connected** button to the Intelligence Portal toolbar, but it does not enable users to query your external data source. To make Connector Creator useful, you must upload the configuration file that defines the queries that users can run.

About this task

The configuration database for Connector Creator stores a single set of configuration settings at any given time. Any upload operation that you perform overwrites any previous settings. As a result, it is easy to create your configuration incrementally, and to test modifications regularly.

To aid debugging, the commands for uploading and downloading the configuration file both support an `"-lp"` option that enables Java logging. The value that you specify is the path to a configuration file that looks like this example:

```
.level=INFO
java.util.logging.FileHandler.limit=50000
java.util.logging.ConsoleHandler.formatter=java.util.logging.SimpleFormatter
handlers=java.util.logging.ConsoleHandler, java.util.logging.FileHandler
java.util.logging.FileHandler.count=1
java.util.logging.FileHandler.formatter=java.util.logging.SimpleFormatter
java.util.logging.ConsoleHandler.level=ALL
java.util.logging.FileHandler.pattern=LogFile
```

Note: This is standard Java logging, and there are many online resources that describe how to create and edit configuration files.

Procedure

1. On the i2 Analyze server, open a command prompt and navigate to the `scripts` directory of the i2 Analyze deployment toolkit.

2. If you have previously uploaded a configuration file, run the following command to download it:

   ```
   setup -t downloadCcConfig -c ExistingConfig.xml
   ```

   By performing this step, you ensure that you can revert to a known configuration in the event that your new file does not work correctly.

   Note: In some circumstances, the downloaded file can contain empty elements that were not present in the original configuration. To prevent this behavior from affecting you, search the file for empty `<queries/>` and `<select/>` elements, and delete any that you find.
3. Run the following command to upload your new Connector Creator configuration file to the configuration database:

   ```
   setup -t uploadCcConfig -c NewConfig.xml
   ```
4. Restart the HTTP server, and then open the Intelligence Portal in your web browser.
   When you click **Connected**, the user interface contains information about the data source to which you provided access, and the queries that users can run.

What to do next

Verify that everything in the user interface that relates to Connector Creator behaves as you expect, and that you can run the queries and interact with their results. If you experience problems, you can examine the `console.log` file in the `deploy\wlp\usr\servers\onyx-server` directory of your i2 Analyze installation, or the additional log files in the `Connected`, `onyx-services-ar`, and `messages` sub-directories of that location.

When you are satisfied with the functionality that you have enabled, you can return to the configuration file and add more queries, more resources, or more data sources to Connector Creator.

The Connector Creator configuration file

To make an external database available through Connector Creator, you must provide a description of the data that you want to access. The Connector Creator configuration file describes and maps the data to the XML schema definition that you generate from your i2 Analyze schema.

In outline, the XML file that maps the data from the external database and specifies the queries that users can run has the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<configuration>
   <dataSource>
      ...
      <securityDimensionValueIds/>
      <mappings>
         <mapping>
            <properties/>
            <propertyGroups/>
            ...
         </mapping>
         ...
         <linkMapping>
            <properties/>
            <propertyGroups/>
            ...
         </linkMapping>
         ...
      </mappings>
      <resources>
         <resource>
            <securityDimensionValueIds/>
            <supportedMappings/>
            <queries>
               <query>
                  <parameters/>
               </query>
               ...
            </queries>
         </resource>
         ...
      </resources>
   </dataSource>
   ...
</configuration>
```

Each `<dataSource>` element represents an external database that is connected through Connector Creator. You can configure the connections to multiple data sources in a single file.

Every `<dataSource>` element contains a `<mappings>` element and a `<resources>` element. The mappings specify how Connector Creator transforms data from an external database into items that users can view and manipulate in i2 Analyze. The resources specify the data that you want to enable users to search, such as database tables or selected columns from one or more tables. The resources also define the queries that users can run against the specified data through the Intelligence Portal.

The <dataSource> element

In the configuration file for Connector Creator, the single <configuration> root element contains one or more <dataSource> elements, each of which specifies an external database. No matter how many data sources Connector Creator is connected to, there is always one configuration file.

The <dataSource> element requires a <name> child element that identifies the database in the Intelligence Portal. Optionally, you can include a <description> element that also provides a description of the external data.

```
<configuration>
   <dataSource>
      <name></name>
      <description></description>
      <specification></specification>

      <securityDimensionValueIds>
         <securityDimensionValueId></securityDimensionValueId>
         ...
      </securityDimensionValueIds>
      <mappings/>
      <resources/>
   </dataSource>
   ...
</configuration>
```

Connector Creator uses the contents of the mandatory <specification> child element to associate this configuration information with an external data source that you added to the i2 Analyze deployment. The contents of the element must match the value of the jndiName attribute of the corresponding <dataSource> element in the server.datasources.xml file.

The <securityDimensionValueIds> element

In i2 Analyze, security dimension values set the access and grant permissions on items, which control the level of access that users receive. In the configuration file, the <securityDimensionValueIds> element contains <securityDimensionValueId> child elements that specify how Connector Creator assigns security dimension values to items that are derived from external data.

In the configuration file, the <securityDimensionValueIds> element can be a child of <dataSource> elements and <resource> elements. In the first case, its contents affect all items that Connector Creator retrieves from that data source. In the second case, the contents of <securityDimensionValueIds> affect items from that resource only.

Note: The effects of these elements are additive. The items from a particular resource receive security dimension values from <securityDimensionValueIds> elements at the local scope, and at the data source scope.

You can specify security dimension values in two ways:

- To give the same security dimension value to all items, you can set the text content of a <securityDimensionValueId> element to the identifier of a security dimension value. For example, <securityDimensionValueId>CON</securityDimensionValueId>.

  Note: You can use this technique at the <dataSource> scope to arrange for all items from the data source to get the same security dimension values as items in the Analysis Repository receive by default. To do so, create a <securityDimensionValueId> element for each value in the

<DefaultItemSecurityTags> setting in toolkit\configuration\fragments\onyx-services-ar\ApolloClientSettings.xml.

- To give different security dimension values to different items, you can use the source attribute to identify a column in the retrieved data that contains security dimension value identifiers. For example, <securityDimensionValueId source="Security_Dimension_1"/>.

  Note: This option is only appropriate for <securityDimensionValueIds> elements that are children of <resource> elements.

If you specify a security dimension value identifier and a source database column, for example <securityDimensionValueId source="Security_Dimension_1">CON</securityDimensionValueId>, then the fixed value applies when source column does not contain a value.

If you specify only a source database column, but the column does not contain a value for a particular item, then another <securityDimensionValueId> element in the configuration file must apply to that item.

The <resources> element

In the Connector Creator configuration file, the <resources> element has child <resource> elements. Each <resource> element specifies a subset of the database to search, and defines the queries that users can run against that data. You create a separate <resource> element for each data subset that you want users to be able to query.

Every <resource> element has a mandatory <specification> child element whose content is an SQL statement. The specification defines the data that each query runs against, and you must enclose the statement in parentheses so that Connector Creator can append query specifications for execution.

For example, the resource might be a database table with the name PERSON, and you might want to allow users to retrieve data from all the columns in the table. In this case, the resource specification might be (*Schema*.PERSON), where *Schema* is the table schema. Alternatively, if you want to retrieve only given names, surnames, and ages from the table, you might set the value to (SELECT FIRST_NAMES, SURNAME, AGE FROM *Schema*.PERSON).

Note: You must use a SQL Alias for each select statement.

To continue the example, a specification for a resource that returns data for creating linked entities might include SQL JOIN statements that combine data from several tables. To retrieve data on people and their vehicles from separate PERSON, OWNS, and VEHICLE tables, the SQL statement might look like this:

```
(SELECT PERSON.PERSON_ID, PERSON.SURNAME, OWNS.BOUGHT,
        VEHICLE.VEHICLE_ID, VEHICLE.MAKE
FROM Schema.PERSON
JOIN Schema.OWNS USING (PERSON_ID)
JOIN Schema.VEHICLE USING (VEHICLE_ID)) AS T
```

If a resource specification returns data for linked entities that have the same type, then the data for both entities must appear in the same set of results. The column names and the aliased column

names that the property mappings specify must match. Here is an example specification for a resource that returns the ages of people that are associated with other people:

```
(SELECT P1.PERSON_ID AS PERSON_ID, P1.AGE as AGE,
        P2.PERSON_ID AS PERSON_ID2, P2.AGE AS AGE2, Schema.PERSON_PERSON.*
FROM Schema.PERSON_PERSON
JOIN Schema.PERSON P1 ON Schema.PERSON_PERSON.FROM_ID = P1.PERSON_ID
JOIN Schema.PERSON P2 ON Schema.PERSON_PERSON.TO_ID = P2.PERSON_ID) AS T
```

Supported mappings

Every <resource> element has a mandatory <supportedMappings> child element contains further <mappingRef> child elements. Each <mappingRef> identifies a mapping that transforms data from the resource into i2 Analyze item data. The contents of a <mappingRef> element must match exactly the value of the mappingID attribute of the required mapping.

```
<dataSource>
   <resources>
      <resource>
         <specification>...</specification>
         <securityDimensionValueIds>
            <securityDimensionValueId>...</securityDimensionValueId>
            ...
         </securityDimensionValueIds>
         <supportedMappings>
            <mappingRef>...</mappingRef>
            ...
         </supportedMappings>

         <queries/>
      </resource>
      ...
   </resources>
</dataSource>
```

A resource that retrieves data for entities of a single type requires one <mappingRef> element. However, a resource that retrieves data for linked entities requires three <mappingRef> elements: one for the link type, and two for the entity types that the link connects. When a link type connects entities of the same type, you must reference both of the entity mappings.

Queries

When a <resource> element has a <queries> child element, the latter contains <query> children that define the search operations that users can run against the data that the resource defines.

Note: If a <resource> does not have a <queries> element, then users cannot run direct queries against it. However, data that the resource retrieves can still appear in the results of "show context" operations.

```
<resource>
   ...
   <queries>
      <query>
         <specification>...</specification>
         <names>
            <name locale="default">...</name>
         </names>
         <descriptions>
            <description locale="default">...</description>
         </descriptions>
         <parameters/>
      </query>
      ...
   </queries>
</resource>
```

Within each <query> element, the value of the mandatory <specification> child element is an SQL statement that refines the specification in the parent resource. When a user runs the query, Connector Creator appends the query specification to the resource specification through a WHERE clause.

For example, you might set the specification to T.AGE=?AGE and create a parameter to enable users to search by age. Alternatively, to create a query with no user-defined parameters that returns the names of 32-year-old people, you might set the specification to T.AGE='32'.

The <query> element has a mandatory <names> child element and an optional <descriptions> child element that identify the query and describe its functionality to users. The text contents of the <name> and <description> elements specify the text that appears in the Intelligence Portal user interface. Both elements have a reserved, mandatory locale attribute that you must set to default in all cases:

Parameters

When you configure a query that enables users to provide search parameters, you must include elements that describe them. The <parameters> element contains <parameter> child elements that

define the search parameters that users can enter, and the fields in the Intelligence Portal where that happens.

```
<query>
   <parameters>
      <parameter parameterName="...">
         <edit>...</edit>
         <labels>
            <label locale="default">...</label>
         </labels>
         <hints>
            <hint locale="default">...</hint>
         </hints>
         <select>
            <options>
               <option value="...">...</option>
               ...
            </options>
         </select>
      </parameter>
      ...
   </parameters>
</query>
```

Each `<parameter>` element has a mandatory `parameterName` attribute that identifies the parameter in the specification of its parent query. The value of the attribute must match the value in the `<specification>` element. For example, if the specification in the query contains a parameter named `?AGE`, then the `parameterName` attribute of the `<parameter>` element must have the value `AGE`.

Note: All the parameters within a query must have different names, but you can use the same parameter name in multiple queries.

If you add no more attributes to the `<parameter>` element, then Intelligence Portal users can enter unconstrained alphanumeric values. However, the element supports optional attributes that allow more control:

Name	Value
`parameterType`	The type of permitted values for the parameter. Legal values for this attribute are `string` for alphanumeric values, `integer` for positive whole numbers, `date` for dates, or `choice` to provide a list of options. **Note:** If you set the type of a parameter to `choice`, you must also provide that parameter with a `<select>` child element.
`inclusiveMin`	For a parameter of type `integer` or `date`, a lower bound on the value that users can enter. For a date, the value must be in the format `yyyy-mm-dd`.
`inclusiveMax`	For a parameter of type `integer` or `date`, an upper bound on the value that users can enter. For a date, the value must be in the format `yyyy-mm-dd`.

Name	Value
required	true to specify that a user must enter a value for this parameter; false otherwise.

To constrain the values of a string parameter, you can use the optional <edit> child element. The text content of the element must be a character data (CDATA) declaration, and the supported syntax is the same as for the first parameter to the JavaScript RegExp constructor, except that the double quotation marks are not required.

Note: The <edit> element does not support the JavaScript global search modifier, but you can emulate it by using a wildcard in the form of an asterisk (*) at the end of an expression. For example, to match any number of digits, set the contents of the <edit> element to <![CDATA[\d*]]>. To configure case-insensitivity, you must specify both uppercase and lowercase character ranges in the expression. For example, <![CDATA[\b[Ii][Bb][Mm]]]> matches the string "IBM" in any combination of uppercase and lowercase letters.

If you set parameterType="choice" on the <parameter> element, then a <select> child element is mandatory. The <select> element in turn has a mandatory <options> child. Inside <options>, each <option> element has a value attribute that determines the value of the parameter when a user selects the option. The text content of the <option> element appears in the Intelligence Portal.

Finally, the <parameter> element has optional <labels> and <hints> children that enable you to specify a label and provide a tooltip for the parameter in the Intelligence Portal. These elements behave like the <names> and <descriptions> children of the <query> element. In particular, locale="default" is mandatory on <label> and <hint> elements.

The <mappings> element

The <mappings> element in the Connector Creator configuration file determines how the data in query results turns into property values in i2 Analyze. To make the most of external data, it is important to get the mappings right.

To work out what the mapping to an item type in the i2 Analyze must look like, you start with the schema4.xsd file that you created in "Creating the Connector Creator configuration file" on page 39. For each item type, the important element is the one whose name attribute is *TypeName*_CardContent. This element describes the properties that items of type *TypeName* can have. For example, here is the start of a definition for an item type named Person:

```
<xs:complexType name="Person_CardContent">
   <xs:sequence>
      <xs:element ... />
      ...
   </xs:sequence>
</xs:complexType>
```

In schema4.xsd, the <xs:element> children of the complex type have name attributes whose format is *TypeName_PropertyName*. These are the names that you use in the mapping. If the type attribute of the

same element indicates a simple data type, then the element represents a property in the schema. If the `type` attribute references a further complex type, then the element represents a property group.

```
<xs:complexType name="Person_CardContent">
   <xs:sequence>
      <xs:element name="Person_FullName" type="Person_FullName"
                  minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Person_DateofBirth" type="xs:dateTime"
                  minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="Person_FullName">
   <xs:sequence>
      <xs:element name="Person_First_Given_Name" type="xs:string"
                  minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Person_FamilyName" type="xs:string"
                  minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
```

In the Connection Creator configuration file, the <mappings> element contains <mapping> and <linkMapping> child elements that map the data in the external data source to entities, links, and properties. Both children have a mandatory `mappingID` attribute that specifies a unique identifier for the mapping within the configuration file. This is the identifier that you provide to a <mappingref> element in a <resource> definition.

```
      <mappings>
         <mapping mappingID="personsMapping">
            ...
         </mapping>
         ...
      </mappings>
```

Each <mapping> element has two mandatory child elements. The contents of the <targetName> element match the name of the schema type that the mapping is for (that is, the *TypeName* part of *TypeName*_CardContent). The contents of the <itemIdentifier> element specify the name of the column in the external data that contains unique identifiers for the items in the search results.

Note: <targetContainerName> is an optional element that can specify the name of the container element for search results of the same type. Usually, the container name is the same as the item type name, but with an "s" appended. In these circumstances, you do not need to include the element.

```
<mappings>
   <mapping mappingID="personsMapping">
      <itemIdentifier>PER_ID</itemIdentifier>
      <targetName>Person</targetName>
      <targetContainerName>Persons</targetContainerName>
      <propertyGroups>
         ...
      </propertyGroups>
      <properties>
         ...
      </properties>
   </mapping>
   ...
</mappings>
```

Each <linkMapping> element has the same child elements as <mapping>, plus four extra, mandatory ones for information that is specific to links:

```
   <linkMapping mappingID="employment">
      <itemIdentifier>PER_ORG_ID</itemIdentifier>
      <fromEndItemIdentifier>PER_ID</fromEndItemIdentifier>
      <toEndItemIdentifier>ORGID</toEndItemIdentifier>
      <strength>Confirmed</strength>
      <direction>NONE</direction>
      <targetName>Employment</targetName>
      <targetContainerName>Employments</targetContainerName>
      <propertyGroups>
         ...
      </propertyGroups>
      <properties>
         ...
      </properties>
   </linkMapping>
```

<fromEndItemIdentifier> and <toEndItemIdentifier> have contents that specify the names of the external data columns that contain unique identifiers for the entities at each end of the link. <strength> and <direction> have contents that specify how certain the link is, and the direction of the relationship that it represents. Confirmed and NONE are common values to use here.

Property groups and properties

If you identify data in your query results that maps to the values of simple properties in the i2 Analyze schema, then you can populate the <properties> element of your <mapping> with <property> children.

Every <property> element has two child elements: <name> and <source>. The contents of <name> must match the name attribute of an <xs:element> in the schema that has a simple data type, as described

at the start of the topic. The contents of <source> must the name of the column in the query results that contains values for this property.

```
<mapping mappingID="personsMapping">
   ...
   <propertyGroups>
      <propertyGroup>
         <name>Person_FullName</name>
         <properties>
            <property>
               <name>Person_First_Given_Name</name>
               <source>GIVEN</source>
            </property>
            <property>
               <name>Person_FamilyName</name>
               <source>FAMILY</source>
            </property>
         </properties>
      </propertyGroup>
   </propertyGroups>
   <properties>
      <property>
         <name>Person_DateofBirth</name>
         <source>DOB</source>
      </property>
   </properties>
</mapping>
```

If you identify data in your query results that maps to properties in the i2 Analyze schema that appear in property groups, then you can populate the <propertyGroups> element of the <mapping>. Each <propertyGroup> element has a mandatory <name> child whose contents must match the name attribute of an <xs:element> in the schema that references a complex data type. <propertyGroup> elements also have a <properties> child that obeys the same rules as the element with the same name that is a direct child of <mapping> or <linkMapping>.

Connector Creator example files

The documentation for Connector Creator includes a pair of example files that you can use to experiment with the technology. The files enable you to populate and query a sample database according to the procedures that the previous topics describe.

example.txt contains an SQL script that creates a small group of database tables and populates them with some sample data.

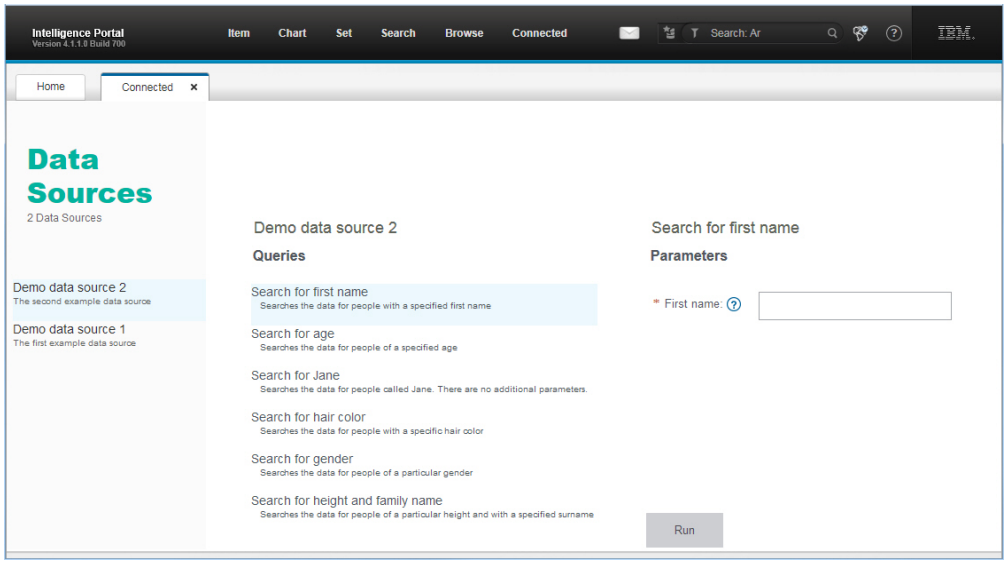
configuration.xml is a Connector Creator configuration file that presents searches over the sample data to Intelligence Portal users. The file also maps search results to be compatible with the default i2 Analyze law enforcement schema, and the default security schema.

In their supplied form, the example files require an empty database named "Example" that they populate and search. Some of the SQL statements in the configuration file assume that the RDBMS is DB2, but it is not difficult to convert them to other dialects if you want to.

To use the example files, assuming that you already installed Connector Creator:

1. Run the script in example.txt to create and populate tables in the example database.

2. Add the example database as a data source in i2 Analyze.
3. Upload the Connector Creator configuration file.
4. Open the Intelligence Portal, and click **Connected** in the menu bar.



Replacing the i2 Analyze schema

An important part of converting an example deployment to a live deployment, is the modification or replacement of the i2 Analyze schema. You replace the schema so that it models the data that you want to analyze.

The i2 Analyze deployment toolkit contains four pairs of example i2 Analyze schemas and charting schemes:

Law Enforcement

The law enforcement schema deals with criminal activity. It contains entity and link types that are designed to track connections within criminal networks.

Commercial Insurance

The commercial insurance schema deals with fraud in a commercial setting. It contains entity and link types that are designed to track financial transactions such as credit card payments and insurance claims.

Military

The military schema helps with military intelligence tracking. It contains entity and link types that target military operations.

Signals Intelligence

The signals intelligence schema focuses particularly on the cellphones and cell towers that are involved in mobile telecommunications, and on the calls that take place between them.

Note: Some of the example schemas and charting schemes are provided in multiple languages. These versions are located in a directory that is named with the language code inside the `configuration\examples\schemas` directory.

Depending on the deployment pattern you chose, your example deployment has either the law enforcement or the signals intelligence schema. Unless your data is very close to those examples, the right approach is to start again from scratch.

Note: If your data is close to the example schema, then you can modify the schema instead of replacing it, and there is slightly less for you to do. You can follow the procedure in Modifying the i2 Analyze schema.

One of the consequences of replacing or significantly modifying the schema is that it becomes incompatible with any data in your deployment, see "Clearing data from the system" on page 149. As a result, these activities require you to clear the data from i2 data stores. The other requirements depend on your deployment pattern.

For the different example configurations that IBM provides, the procedures for replacing the i2 Analyze schema are as follows:

- If your deployment contains only the Analysis Repository, then redeploying with your new schema (and an accompanying charting scheme) is enough. See "Deploying i2 Analyze" on page 150.
- If you deployed the Information Store with the Opal services, then you also need to replace the configuration file that governs what filters users see during Quick Search and Visual Query operations. See "Setting up search results filtering" on page 119.
- If you deployed the Information Store with the Onyx services, then you also need to replace the Cognos reports and mapping files that control how users interact with search results. See "Configuring the i2 Analyze Onyx search" on page 139.

Important: Replacing the schema is about the most significant change you can make to a deployment of Enterprise Insight Analysis. Aim to do it soon after a successful example deployment, and long before a production deployment. Large changes to an i2 Analyze schema when users and data depend on it are hard to manage.

You can learn more about deploying a custom schema at the IBM Security Learning Academy.

Creating an i2 Analyze schema and charting schemes

The supplied i2 Analyze schemas are suitable for production deployments of Enterprise Insight Analysis only when the target environment is close to one of the example deployments. Usually, you must prepare for production deployment by creating your own i2 Analyze schema and your own versions of the other configuration files that depend upon it.

The deployment toolkit includes example schemas that target particular domains: commercial insurance, law enforcement, military intelligence, and signals intelligence.

- If one of the supplied schemas matches your requirements, you can use it without modification. Even so, depending on the deployment pattern, you might need to create or edit some of the other Enterprise Insight Analysis configuration files.
- Alternatively, one of the supplied schemas might contain some of the entity and link types that you need. If you can meet your requirements by expanding or modifying the existing types, then you can edit the schema to make it fit for purpose.
- If none of the supplied schemas is appropriate, then you can create your own schema that precisely models the data in your organization.

When you create a schema, you must also create one or more *charting schemes* to accompany it. These separate files specify how entities and links and their properties appear when they are visualized on charts. To provide users with more flexibility in their investigations, you can visualize data in different ways by creating multiple charting schemes.

The deployment toolkit includes charting schemes that match the example schemas. You can update these schemes to match any modifications you make to the example schemas; or else you can create charting schemes that match a custom schema.

After you create a schema and its corresponding charting schemes, you can deploy it into a test environment for evaluation. Creating an effective schema involves several cycles of reviewing and testing before you use it in a production environment.

After a schema enters production, the changes that you can make to it are strictly limited. Therefore, it is important to ensure that the schema meets the requirements of the organization before it goes live.

You can view, edit, create, and upload schemas with Schema Designer. For instructions on creating and editing an schema, see the Schema Designer documentation.

Specifying the i2 Analyze schema and the charting scheme

Developing an i2 Analyze schema and a charting scheme for an Enterprise Insight Analysis deployment is an iterative process. You can expect to modify the schema several times as you build towards the production version. As you do so, you must change other parts of the system too.

Before you begin

Each time you come to this task, you have an i2 Analyze schema that is different from the one that currently applies to the deployment. Either you created a schema, or you edited an existing schema. Now you want to apply the schema to the deployment and test it.

About this task

This procedure contains an outline of the steps to follow as you develop a schema for your deployment of i2 Analyze. In reality, you are likely to follow some of the steps on each iteration of your design. At the beginning of the process, when you create or copy the schema and charting scheme files, you must specify the new files in the deployment toolkit. As you develop the schema, you might need to clear data from the system, or modify your sample data, or edit another part of the configuration that depends on the schema.

Procedure

Much of the work in developing the i2 Analyze schema and the other files that depend on it lies in making them consistent with each other. One approach is to start with a schema that contains only a few of the types that you plan to create. Then, get an i2 Analyze deployment working that uses those types. After that, you can add more types to a system that you know to be sound.

1. If you created an i2 Analyze schema and charting scheme, copy the files to the `configuration\fragments\common\WEB-INF\classes` directory of the deployment toolkit.
2. If you changed the file names of the i2 Analyze schema and charting scheme, set the i2 Analyze schema and charting scheme that the deployment uses.
   a) Using a text editor, open the `ApolloServerSettingsMandatory.properties` file in the same directory as the schema files.
   b) Set the values of the `SchemaResource` and `ChartingSchemesResource` properties to the names of your schema and charting scheme.
   c) Save and close the settings file.
3. If you made anything other than additive changes to the schema, then clear the data, the search index, and the database from the system.

   See “Redeploying and resetting i2 Analyze” on page 148.

4. If your deployment includes the Opal services, create and configure facets to match the item and property types that you added or changed in the i2 Analyze schema.

   See “Setting up search results filtering” on page 119.
5. If your deployment includes the Onyx services and the Information Store, edit the Cognos mappings to match the schema.

   See “Configuring the i2 Analyze Onyx search” on page 139.
6. If you have sample data that does not exercise or is no longer compatible with the new version of the schema, update it.
7. Redeploy i2 Analyze.

What to do next

Repeat the procedure until your i2 Analyze schema fulfills the needs of your organization. When the schema reaches that point, you can go on to finalize the charting scheme and develop the security schema.

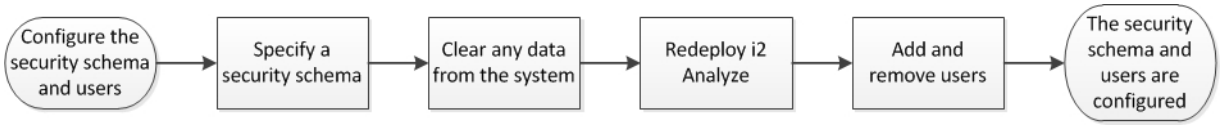
Configuring the security schema and users

Example deployments of i2 Analyze have security schemas that allow the software to work, but are not suitable for production purposes. Before you go live, you must develop a security schema that meets the needs of your organization.

An i2 Analyze security schema defines the security dimension values that you can assign to items and records in i2 data stores, and the security permissions that you assign to groups of users.

The fact that the data stores contain dimension values means that significant changes to the security schema can require you to clear and reingest or repopulate data. This behavior is reasonable while you develop your deployment, but not after you go live with it. As a result, it is important to refine the security schema before you put it into production. After you go live, the changes that you can make are much more limited.

The following diagram shows the workflow for configuring security during development of your i2 Analyze deployment:



In addition to the security schema, a separate file can be created that allows you to control access to specific features and commands.

Security schemas

An i2 Analyze security schema defines the security dimensions that exist in a deployment, and the dimension values that can be assigned to items and records. A security schema also defines the permissions that i2 Analyze users can receive.

Every deployment of i2 Analyze has a security schema whose contents reflect local requirements. It is the responsibility of the deployer to ensure that the security schema is appropriate for the environment where it is used. Often, the security dimensions map to security classifications that exist in the organization.

Before you create a security schema, it is important to understand the relationship between the security model and the security schema. For more information, see IBM i2 Analyze security model .

Security dimensions

A security schema defines access security dimensions and grant security dimensions separately. Although they have the same structure, access and grant security dimensions are distinct from each other. In a security schema, dimensions and dimension values must have identifiers that are unique across the whole schema.

Security permissions

A security schema defines security permissions by user group, and then by dimension. For a particular user group, the schema identifies one or more dimensions for which membership of that group affects access rights. For each identified dimension, the schema contains a list of security permissions.

It is not necessary for the security schema to define permissions for every user group in the organization. Similarly, it is not necessary for the permissions within any particular dimension or group to set a security level for every possible dimension value. The completeness of the schema is judged at run time when the security level of a particular user for a particular item or record is calculated.

Security schema definitions

An i2 Analyze security schema is an XML file with a relatively simple structure. Security dimensions and security permissions are defined in separate sections of the file.

In outline, the `<SecuritySchema>` root element of a security schema contains child elements for the dimension and permission definitions:

```
<SecuritySchema>
    <SecurityDimensions Id="" Version="">
        <AccessSecurityDimensions>
            <Dimension ...>
                <DimensionValue ... />
                ...
            </Dimension>
            ...
        </AccessSecurityDimensions>

        <GrantSecurityDimensions>
            <Dimension ...>
                <DimensionValue ... />
                ...
            </Dimension>
            ...
        </GrantSecurityDimensions>
    </SecurityDimensions>

    <SecurityPermissions>
        <GroupPermissions ...>
            <Permissions ...>
                <Permission ... />
                ...
            </Permissions>
            ...
        </GroupPermissions>
        ...
    </SecurityPermissions>
</SecuritySchema>
```

The `<SecurityDimensions>` element has attributes for the `Id` and `Version` of the schema. If you modify any part of the security schema, retain the identifier but increment the version number. In this way, you ensure that all i2 data stores and services are informed of the changes.

In a valid security schema, the `<AccessSecurityDimensions>` and `<GrantSecurityDimensions>` elements must be present, and there must be at least one `<GroupPermissions>` element inside `<SecurityPermissions>`.

Security dimension definitions

Security dimensions are defined in an i2 Analyze security schema file as children of the mandatory <AccessSecurityDimensions> and <GrantSecurityDimensions> elements. A valid security schema defines at least one access and one grant security dimension.

The syntax for defining a security dimension does not depend on whether it is an access or a grant dimension. The structure of the XML is always the same. The following example shows a simple, complete <Dimension> element:

```
<Dimension Id="SD-SC"
           DisplayName="Security Classification"
           Description="The security classification of this information"
           Ordered="true">
    <DimensionValue Id="TOP" DisplayName="Top Secret" Description="Top Secret" />
    <DimensionValue Id="RES" DisplayName="Restricted" Description="Restricted" />
</Dimension>
```

The attributes of the <Dimension> element affect how the values in the security dimension are interpreted.

Attribute	Description
Id	A unique identifier that is used to distinguish this security dimension throughout the system.
DisplayName	A name that identifies this dimension to the user in the Intelligence Portal, for example.
Description	A more detailed description of this security dimension that provides more information to the user. In the Intelligence Portal, the description is used as a tooltip.
Ordered	Indicates whether the values in this dimension form a descending sequence in which each value supersedes the values below it. Marking this dimension as Ordered="true" means that a user who has access rights to "Top Secret" data implicitly has the same access rights to "Restricted" data as well. For a dimension in which Ordered="false", there is no such implication, and access rights must be assigned explicitly for each dimension value.

The Id, DisplayName, and Description attributes of <DimensionValue> elements have the same purpose and meaning as the <Dimension> attributes with the same names. The identifiers of dimension values must be unique within the dimension that defines them.

Important: After you deploy i2 Analyze, the changes that you can make to security dimensions are limited. You cannot add or remove dimensions, or remove dimension values. You can only add values to existing dimensions. For this reason, you must understand the requirements of your organization before you deploy i2 Analyze in a production environment.

Security group permission definitions

In an i2 Analyze security schema, the mandatory <SecurityPermissions> element contains one or more <GroupPermissions> elements. Each <GroupPermissions> element defines the security levels that

users in a particular group receive for items and records with particular dimension values in an i2 Analyze deployment.

The syntax for defining the security permissions for user groups enables membership of one group to convey permissions across several dimensions, and allows different groups to convey different permissions for the same dimensions. The following example shows how to structure `<GroupPermissions>` elements inside the `<SecurityPermissions>` element:

```
<SecurityPermissions>
    <GroupPermissions UserGroup="Clerk">
        <Permissions Dimension="SD-SC">
            <Permission ... />
        </Permissions>
        ...
    </GroupPermissions>
    <GroupPermissions UserGroup="Manager">
        <Permissions Dimension="SD-SC">
            <Permission ... />
            ...
        </Permissions>
        <Permissions Dimension="SD-IT">
            <Permission ... />
            ...
        </Permissions>
        ...
    </GroupPermissions>
    <GroupPermissions UserGroup="Security Controller">
        <Permissions Dimension="SD-GA">
            <Permission ... />
        </Permissions>
    </GroupPermissions>
</SecurityPermissions>
```

The value of the `UserGroup` attribute of each `<GroupPermissions>` element must match the name of a group of i2 Analyze users.

The value of the `Dimension` attribute of each `<Permissions>` element must match the identifier of one of the dimensions that is defined in the first part of the schema.

It is normal for `<Permissions>` elements for the same dimension to appear in more than one `<GroupPermissions>` element:

- Users who are members of one group but not the other can receive different access levels on items and records that have the same dimension values.
- When users are members of more than one group, `<Permissions>` elements for the same dimension are combined before any access level calculation takes place.

In many deployments of i2 Analyze, there is one grant security dimension that contains one dimension value. By referring to the grant security dimension from a single `<GroupPermissions>` element, you can arrange for grant access to be in the hands of users who are members of a dedicated group.

Important: You can add and remove `<GroupPermissions>` elements from a deployed security schema, if the resulting system continues to obey the rules of i2 Analyze. In particular, it must remain possible for all users to get an access level that is not "none" for at least one value in every access dimension.

Security permission definitions

The security permission definitions in an i2 Analyze security schema each associate a single dimension value with a single security level. The definitions can be simple because of the additional context that their location in the security schema file provides.

The `<Permission>` elements that define security permissions always appear inside `<Permissions>` elements, which in turn always appear inside `<GroupPermissions>` elements.

```
<GroupPermissions UserGroup="Manager">
    <Permissions Dimension="SD-SC">
        <Permission DimensionValue="TOP" Level="READ_CLOAKED" />
        <Permission DimensionValue="RES" Level="UPDATE" />
    </Permissions>
    <Permissions Dimension="SD-IT">
        <Permission DimensionValue="HUMINT" Level="READ_ONLY" />
    </Permissions>
</GroupPermissions>
```

It is possible, and often desirable, for identical `<Permission>` elements to appear in different locations in an i2 Analyze security schema. The effect of a security permission definition depends entirely on its position in the file.

Important: Like the `<GroupPermissions>` elements that contain them, you can add and remove `<Permissions>` and `<Permission>` elements from a deployed security schema, if the resulting system does not break the rules of i2 Analyze.

Creating a security schema

Every deployment of i2 Analyze requires a security schema that encapsulates the security model for that deployment. The easiest way to create a security schema is to start from the example that IBM provides with the platform.

Before you begin

Before you create the XML security schema file, you must design the security model for your deployment of i2 Analyze. In particular, you must identify or create the user groups to which security permissions are assigned.

When you deploy i2 Analyze, the group names in your security schema must match the names of user groups in the user repository.

About this task

An i2 Analyze security schema contains definitions of security dimensions and security permissions. When you create a security schema, you define the dimensions and dimension values first, and then define the security permissions that refer to them.

Procedure

1. Navigate to the directory in the deployment toolkit that contains the example security schema: `toolkit\configuration\examples\security-schema\example-dynamic-security-schema.xml`.
2. Make a copy of the `example-dynamic-security-schema.xml` file, give it an appropriate name, and then open it in an XML editor.

3. Edit the contents of the <AccessSecurityDimensions> element so that it contains a <Dimension> element for each category that your deployment uses to determine access rights to items and records in i2 data stores.
4. Edit the contents of the <GrantSecurityDimensions> element so that it contains a <Dimension> element whose value you can use to convey grant access rights.
5. Edit the contents of the <SecurityPermissions> element:
   a) Add or modify <GroupPermissions> elements so that they reflect all the user groups to which you assign security permissions. The group names in your security schema must match the names of user groups in the user repository.
   b) Within each <GroupPermissions> element, add or modify <Permissions> elements to indicate which dimensions are affected by membership of each user group.
   c) Within each <Permissions> element, add or modify <Permission> elements to assign security levels to items and records that have particular dimension values.

      If your deployment of i2 Analyze includes an Analysis Repository but no Information Store, there are four permitted values for the Level attribute of the <Permission> element:

      - NONE
      - READ_CLOAKED
      - READ_ONLY
      - UPDATE

      If your deployment includes an Information Store, then READ_CLOAKED is not a permitted value, and a security schema that contains it is invalid.
6. Increment the value of the Version attribute of the <SecurityDimensions> element.
7. Save the completed security schema to the configuration\fragments\common\WEB-INF\classes directory in the deployment toolkit.

What to do next

Modify the deployment toolkit to specify that i2 Analyze uses the new security schema, and then redeploy the platform and conduct your tests. If necessary, iterate over these steps again until you have the security schema that you need.

Specifying a security schema

The security schema is a key component of an i2 Analyze deployment, and configuring it correctly is an important part of the development process. Replacing or making significant changes to the security schema requires you to clear and repopulate the data stores in your deployment.

About this task

One of the features of i2 Analyze security is that items in the Analysis Repository and records in the Information Store must have dimension values from every dimension in the security schema. As a result, it is not possible to add security dimensions or remove dimension values without invalidating the data in the data stores. IBM recommends that you build and refine the security schema before your deployment goes into production use. See Understanding IBM i2 Analyze: Security model for more information about the security model and the security schema.

Procedure

After you perform an example deployment of i2 Analyze, use the following procedure to replace the security schema with one that you created or modified.

Note: For a description of how to make small modifications to the security schema in a production deployment of i2 Analyze, see Modifying the security schema.

1. In Windows Explorer, navigate to the `configuration\fragments\common\WEB-INF\classes` directory in the deployment toolkit.
2. Open the `ApolloServerSettingsMandatory.properties` file in a text editor.
3. Set the value of the `DynamicSecuritySchemaResource` property to the name of your new or modified security schema.
4. Save and close the properties file.
5. Follow the instructions in "Clearing data from the system" on page 149 to remove all the data from the data stores in the i2 Analyze deployment.
6. Follow the instructions in "Deploying i2 Analyze" on page 150 to deploy your new or modified security schema and restart the system.

What to do next

After you change the security schema, you might need to change the dimension values that records and items receive when they enter the system. After those changes, add some data to the system, and verify that users see the behavior that you intended. Iterate over the process of modifying and replacing the schema as many times as you need.

Setting default dimension values for Opal

In a deployment with the Opal services, when users create Information Store records in Analyst's Notebook Premium, i2 Analyze applies a default set of dimension values to the record. Similarly, when the Information Store ingests data, the created records receive dimension values during the process.

About this task

You might change the default security dimension values that are applied to records for different reasons:

- If you change the dimensions or dimension values in your security schema.
- If the security requirements of your deployment change.

Regardless of your reason for changing the default security dimension values, or when the values are applied, each record must have at least one dimension value from each security dimension.

Procedure

1. When analysts create Information Store records in Analyst's Notebook Premium, the records receive default security dimension values. To change the set of security dimension values that records receive by default:
   a) In a text editor, open the `configuration\fragments\common\WEB-INF\classes\ApolloServerSettingsMandatory.properties` file from the deployment toolkit.
   b) Update the `DefaultSecurityDimensionValues` property with the identifiers of security dimension values that you want to be applied by default.
   For example, `DefaultSecurityDimensionValues=CON,OSI,HI`.

   In a standard deployment of i2 Analyze, a supplied implementation of the `DefaultSecurityDimensionValuesProvider` that applies dimension values from the `DefaultSecurityDimensionValues` property is used. You can create your own implementation by using i2 Analyze Developer Essentials.
2. When data is ingested into the Information Store, the records that it contains receive their security dimension values during the ingestion process.

To change the behavior of the ingestion process, update the contents of the `<securityDimensionValues>` element in the mapping file with the identifiers of security dimension values that you want to be applied by default.

For more information about updating the default security dimension values that are applied during ingestion, see Information Store data ingestion.

3. Redeploy and restart your deployment of i2 Analyze.

Setting default dimension values for Onyx

In a deployment with the Onyx services, when a user creates an item in the Analysis Repository, i2 Analyze applies a default set of dimension values. Similarly, when the Information Store ingests data, the created records receive dimension values during the process.

About this task

You might change the default security dimension values that are applied to items and records for different reasons:

- If you change the dimensions or dimension values in your security schema.
- If the security requirements of your deployment change.

Regardless of your reason for changing the default security dimension values, or when the values are applied, each item and record must have at least one dimension value from each security dimension.

Procedure

1. If you have an Analysis Repository, i2 Analyze assigns dimension values to items in several different circumstances. Items receive dimension values when Intelligence Portal users create them. Items also receive dimension values when users search for them through an iBase connector, Connector Creator, or a custom data access on-demand solution.
   a) To change the set of security dimension values that items in the Analysis Repository receive by default:
      1) In an XML editor, open the `configuration\fragments\onyx-services-ar\ApolloClientSettings.xml` file from the deployment toolkit.
      2) Update the `<DefaultItemSecurityTags>` element with the identifiers of security dimension values that you want to be applied by default.
      3) Save and close the file.
   b) If your deployment of i2 Analyze includes an iBase connector, edit the `configuration\environment\iBase\environment.properties` file in the deployment toolkit. Update the `ibase.security.context` setting with the identifiers of security dimension values that you want to be applied by default.
   c) If your deployment of i2 Analyze includes Connector Creator, update the contents of the `<securityDimensionValueIds>` element in the configuration file with the identifiers of security dimension values that you want to be applied by default.
   d) If your deployment of i2 Analyze includes a custom data access on-demand solution, update its behavior to provide the items that it retrieves with the security dimension values that you want to be applied by default.
2. If you have an Information Store, the records that it contains receive their security dimension values during the ingestion process.

To change the behavior of the ingestion process, update the contents of the `<securityDimensionValues>` element in the mapping file with the identifiers of security dimension values that you want to be applied by default.

For more information about updating the default security dimension values that are applied during ingestion, see Information Store data ingestion.

3. Redeploy and restart your deployment of i2 Analyze.

Setting up WebSphere Application Server Liberty profile security

The access levels that each user receives within i2 Analyze are determined by their membership of groups. The names of these groups must match the group permissions elements values that are defined in your security schema.

Before you begin

Ensure that the application server is stopped. To stop the application server, run the following command:

```
setup -t stop
```

About this task

Important: WebSphere Application Server Liberty profile can handle multiple security approaches. Always use an approach that is appropriate to the environment that you are deploying into. For information on WebSphere Application Server Liberty profile security approaches, see Securing the Liberty profile and its applications.

The following rules apply:

1. You must create groups in the WebSphere Application Server Liberty profile user registry whose names exactly match the `UserGroup` attribute of the group permissions elements in the security schema.
2. You must ensure that every user is a member of enough groups such that they are assigned a dimension value and level from each access security dimension. A user does not require a mapping to a grant security dimension.

To illustrate these rules, consider that the example security schema defines the following dimensions and groups:

Group Permissions `UserGroup` value	**Group Permissions for Dimension**	**Dimension values and level**
Analyst	Security Compartment	Human Informants - update, Open Source Intelligence - read_only
Clerk	Security Compartment	Open Source Intelligence - update
Controlled	Security Level	Controlled - update
Unclassified	Security Level	Controlled - update, Unclassified - update
Security Controller	Grant Access	Security Controller - update

To map to this security schema, the user group values in the table must match with the user groups in the user repository.

Each user in this deployment must be in either of the "Analyst" or "Clerk" groups, and either of the "Controlled" or "Unclassified" groups.

Every deployment must contain an account that is associated with the administrator role. You can create a group in the user repository named "Administrator", or you can change the value of the `security.administrator.group` property to the name of an existing group in the repository. The `security.administrator.group` property is in the `environment-advanced.properties` file for each application, in the `toolkit\configuration\environment\`*`application`* directory. When an i2 Analyze user is a member of this group, they can access administrative features.

The following process is an approach to security in WebSphere Application Server Liberty profile that uses a basic user registry.

Procedure

1. Create the users and groups in WebSphere Application Server Liberty profile for each of the group permissions elements in the security schema.
   a) In an XML editor, open the `user.registry.xml` file. You can find this file in the `C:\IBM\i2analyze\deploy\wlp\usr\shared\config` directory of your WebSphere Application Server Liberty profile installation.
   b) Use the following template to add your users and groups to the `user.registry.xml` file as the first child of the `<server>` element:

```
<basicRegistry id="basic" realm="WebRealm">
    <user name="" password="" />
    <group name="">
        <member name="" />
    </group>
</basicRegistry>
```

   Use the following information to populate the template:

   - There is a `<user>` element for each user of the system. The `<user>` element's `name` and `password` attributes must be populated for that user.
   - There is a `<group>` element with a `name` attribute that matches the name of each security dimension in the security schema.
   - The `<group>` elements are populated by `<member>` elements. For a user to be a member of a group, a `<member>` element's `name` attribute must match that user's `name` attribute.

   If you are using the example deployment, the user `Jenny` is a member of each group.

In the following example `user.registry.xml`, the users `Analyst1`, `Clerk1`, and `Demo` have been added into a subset of the groups. If you use the following example, log in as these users to see the different permission levels of each group:

```
<basicRegistry id="basic" realm="WebRealm">
        <user name="Jenny" password="{xor}FToxMSY="/>
        <user name="Analyst1" password="{xor}Lz4sLCgwLTs=" />
        <user name="Clerk1" password="{xor}Lz4sLCgwLTs=" />
        <group name="Analyst">
            <member name="Jenny"/>
            <member name="Analyst1"/>
        </group>
        <group name="Clerk">
            <member name="Jenny"/>
            <member name="Clerk1"/>
        </group>
        <group name="Controlled">
            <member name="Jenny"/>
            <member name="Analyst1"/>
        </group>
        <group name="Unclassified">
            <member name="Jenny"/>
            <member name="Clerk1"/>
        </group>
        <group name="Security Controller">
            <member name="Jenny"/>
        </group>
        <group name="Administrator">
            <member name="Jenny"/>
        </group>
  </basicRegistry>
```

The passwords are encoded by the WebSphere Application Server Liberty profile security utility.

2. Use the WebSphere® Application Server Liberty profile `securityUtility` command to encode the password for each user.
   a) Navigate to the `bin` directory of your WebSphere Application Server Liberty profile deployment that is configured by the deployment toolkit. By default WebSphere Application Server Liberty profile is deployed in the `C:\IBM\i2analyze\deploy\wlp` directory.
   b) In a command prompt, run the following command:

   ```
   securityUtility encode password
   ```

   The encoded password is displayed in the command line. Record the encoded password, including the `{xor}` prefix, and use the encoded password as the password in the `user.registry.xml` file.

   For more information about using the security utility, see securityUtility command.
3. Save and close the file.

4. To start the application server, open a command prompt and navigate to `toolkit\scripts`. Then, run the following command:

```
setup -t start
```

5. Start, or restart, the HTTP server that hosts the reverse proxy.

What to do next

To test that your changes have worked, log in to i2 Analyze as one of the users that you added to the user registry.

Controlling Access to features

In addition to access to records, you can also control access to features or types of command. To restrict access to features, you need to specify a command access control file.

You can use a command access control file to specify the permissions to assign to each user group that you specified in the security schema. The command access control file can be used to set permissions that apply to i2 Analyze and, if applicable, to other custom client services.

Note: By default access control is not enabled, allowing all authenticated users access to all features. After you have set command access control, you can revert to this default state by ensuring that the `CommandAccessControlResource` property in the `toolkit\configuration\fragments\opal-services-is\WEB-INF\classes\DiscoServerSettingsCommon.properties` has no value.

The following permissions apply to the features and commands present in i2 Analyze:

i2:Notes

Members of groups that have this permission can create and access notes on records. Without this permission, records do not display the Notes tab, and the contents of any notes are not searchable.

i2:RecordsUpload

Members of groups that have this permission can create and modify records and upload them to the Information Store. Without this permission, records can be searched, and added to charts, but changes to records cannot be shared.

i2:RecordsDelete

Members of groups that have this permission can delete records were originally uploaded using Analyst's Notebook Premium. Without this permission, records can be searched, added to charts, and uploaded, but records cannot be removed from the Information Store.

Note: When you upgrade a system that has access to specific features enabled, ensure that you check for new features that require new permissions. Without updating your file to add these permission, access to the features will be denied to all users.

Setting up example command access control

The `example-command-access-control.xml` file provides a demonstration of command access control that matches the example security schema. If you have an example deployment, you can use this file to test command access control.

Procedure

To enable the command access control example:

1. Copy the `example-command-access-control.xml` file from `toolkit\configuration\examples\security-schema` to `toolkit\configuration\fragments\opal-services-is\WEB-INF\classes`.

2. To specify the example file to be used in `DiscoServerSettingsCommon.properties`:
   a) Using a text editor, open the `toolkit\configuration\fragments\opal-services-is\WEB-INF\classes\DiscoServerSettingsCommon.properties` file.
   b) Specify the example file as the value for the `CommandAccessControlResource` property:

```
CommandAccessControlResource=example-command-access-control.xml
```

   c) Save the file.
3. Follow the instructions in "Deploying i2 Analyze" on page 150 to enable command access control and restart the system.

Setting up command access control to match your environment

You can use command access control to determine the access to commands and features in your deployment. You can create a command access control file to match the specific needs of your deployment.

Before you begin

Before you create the command access control file, you must identify the user groups for use by your deployment. When you deploy i2 Analyze, the group names in your command access control file must match the names of user groups in the security schema.

Procedure

1. Create a custom command access control file.
   a) Navigate to the directory in the deployment toolkit that contains the example security schema: `toolkit\configuration\examples\security-schema\example-command-access-control.xml`.
   b) Make a copy of the `example-command-access-control.xml` file, give it an appropriate name, and then open it in an XML editor.
   c) For each user group in your deployment, create a `CommandAccessPermissions` element that specifies the permissions you would like to grant.
   For example:

```
<CommandAccessPermissions UserGroup="Analyst">
        <Permission Value="i2:RecordsUpload" />
        <Permission Value="i2:RecordsDelete" />
    </CommandAccessPermissions>
```

   Note: For permissions that apply to all user groups, to prevent the need to give the permission to each group individually, you can use a wildcard character:

```
<CommandAccessPermissions UserGroup="*">
        <Permission Value="i2:Notes" />
    </CommandAccessPermissions>
```

   d) Save the completed file to the `configuration\fragments\opal-services-is\WEB-INF\classes` directory in the deployment toolkit.
2. To set the command access control file to be used in the deployment:
   a) Using a text editor, open the `toolkit\configuration\fragments\opal-services-is\WEB-INF\classes\DiscoServerSettingsCommon.properties` file.

b) Specify your command access control file as the value for the `CommandAccessControlResource` property.
For example:

```
CommandAccessControlResource=my-command-access-control.xml
```

c) Save the file.

3. Follow the instructions in “Deploying i2 Analyze” on page 150 to enable command access control and restart the system.

Configuring additional security

Depending on the requirements of the deployment, you can configure your deployment to use additional security mechanisms.

Secure Sockets Layer connections with i2 Analyze

Secure Sockets Layer (SSL) technology can be used to establish an encrypted connection between a client and server. You can use SSL to ensure that communication between i2 Analyze components is encrypted.

Depending on your topology and requirements, you can configure SSL for the following connections:

- The client and the HTTP Server
- The HTTP Server and WebSphere Application Server Liberty
- WebSphere Application Server Liberty and the search indexing mechanism
- WebSphere Application Server Liberty and the database management system

The versions of the TLS protocol that are supported by i2 Analyze are TLS V1.1, and TLS V1.2.

The instructions are intended for readers who are familiar with configuring i2 Analyze, securing network connections, and managing SSL key authentication certificates. Refer also to the documentation for the individual components: IBM HTTP Server, IBM WebSphere Application Server Liberty profile, Apache Solr, DB2, or Microsoft SQL Server.

The instructions are based on a sample scenario for a single-server deployment. The instructions use self-signed certificates to demonstrate working SSL configurations. In a production deployment, you must use certificates that are signed by a trusted certificate authority.

The instructions use self-signed certificates to demonstrate working SSL configurations. In a production deployment, you must use certificates that are signed by a trusted certificate authority.

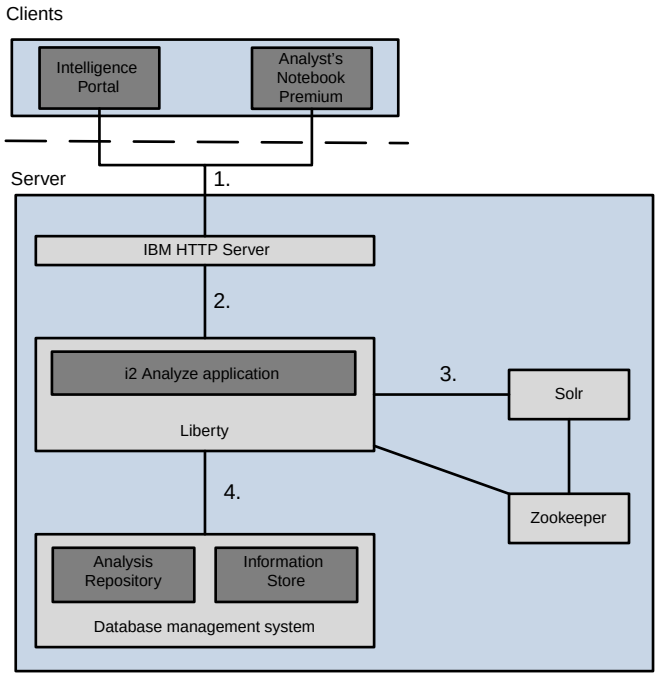
Attention: IBM takes reasonable steps to verify the suitability of i2 Analyze for Internet deployment. However, it does not address lower-level issues such as guarding networks against penetration, securing accounts, protecting against brute force attacks, configuring firewalls to avoid DoS or DDoS attacks, and the like. For your deployment of i2 Analyze, follow industry-standard practices and recommendations for protection of your systems. IBM accepts no liability for the consequences of such attacks on your systems. This information is not intended to provide instructions for managing key databases or certificates.

Connections between components of i2 Analyze

Depending on your deployment pattern, i2 Analyze connections can involve the client, the HTTP server, WebSphere Application Server Liberty, the database management system, and the search indexing mechanism. You can choose to secure connections by using SSL.

The physical architecture and the network topology of an i2 Analyze deployment determine how appropriate it is to use SSL to secure its connections. For example, if two parts of the deployment are on a single machine that you physically control, then the need for SSL to secure the connection between them might be reduced. If your deployment contains a firewall, then you might need to weigh the benefits of inspecting traffic against the benefits of encrypting it.

The following diagram shows the connections in a deployment that you can use SSL to secure:



The numbered connections in the diagram are as follows:

1. The connections between the clients and the HTTP server.
2. The connection between the HTTP server and Liberty.
3. The connection between Liberty and the search indexing mechanism.
4. The connection between Liberty and the database management system.

To secure connection 2, you must first secure connection 1. To secure connection 3, you must secure connections 1 and 2.

Important:

- You can secure the connection to the search indexing mechanism by using SSL only if your deployment uses an Information Store with Opal services. In Opal services Apache Lucene, Solr, and ZooKeeper enable a high standard of search and indexing performance.

- ZooKeeper does not support SSL encrypted communication with clients. To secure the connection to ZooKeeper, you can set up a secure connection by using an alternative method to SSL, such as an SSH tunnel.
- You can secure the connection to the database management system by using SSL only if your deployment uses one of the following options.
  - The Analysis Repository on DB2
  - A Microsoft SQL Server
  - An Information Store on DB2 with the Opal services

SSL certificates for i2 Analyze

SSL communication relies on encryption, keys, and certificates to initiate a secure connection. The certificates are stored in keystore files on the client and the server.

Certificates are exchanged to establish trust during the handshake process that initiates a secure connection. When a certificate is granted through a certificate authority, that certificate can be trusted by the clients or applications that trust certificates that are signed by that authority. A public key certificate that authenticates a server is stored in a keystore file on the server. Trusted certificate authority certificates are stored in the client's truststore file.

As part of the SSL handshake process, certificates are exchanged that are signed by a trusted certificate authority to verify that a certificate is authentic. The environment where you are deploying i2 Analyze might already have a well-defined certificate process that you can use to obtain certificates and populate the required key and trust stores.

The examples in the following procedures use self-signed certificates to demonstrate working SSL configurations. In a production deployment, you must use certificates that are signed by a trusted certificate authority.

If all of the public key certificates are signed by the same certificate authority, then you can add the certificate authority's certificate to each of your truststores. If you have a number of certificates to authenticate trust, you might have to add multiple certificates to your truststores.

In the examples, the self-signed certificate is created in a keystore, exported, and imported to the relevant truststore. When you configure SSL communication between components of i2 Analyze, you must have the required certificates in the correct keystores. In the following procedures, examples commands are provided for creating, exporting, and importing self-signed certificates.

SSL keystores for i2 Analyze

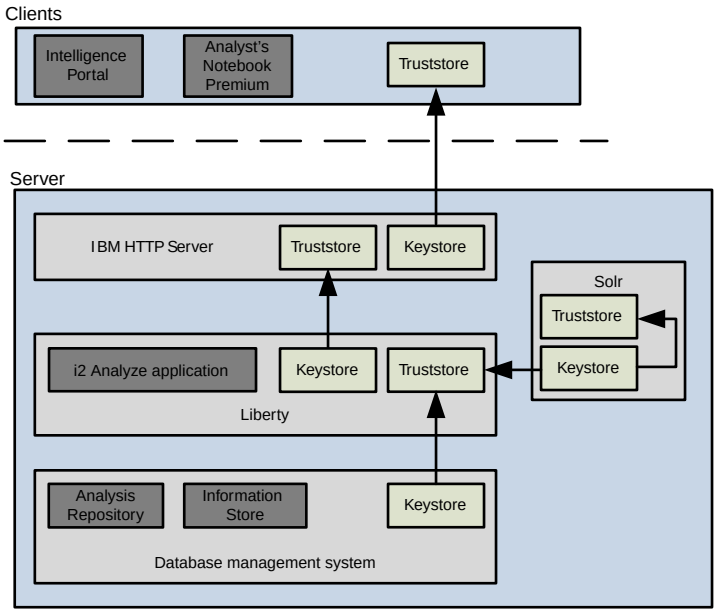
To use SSL to secure a connection between components, i2 Analyze requires Java KeyStore (JKS) or Certificate Management System (CMS) key database files depending on the components. The keystore contains the certificate that acts as the identity of the server and the truststore contains a list of certificates that are trusted by a server.

A JKS file is a repository of security certificates that is used in SSL encryption. In WebSphere Application Server and Apache Solr, a file with extension `.jks` serves as a keystore.

IBM Global Security Kit (GSKit) is a common component that is used by IBM HTTP Server and DB2 for its cryptographic and SSL capabilities. CMS is the native GSKit key database (keystore) that contains certificates. GSKit stores public and private keys and certificates in a key database. A key database consists of a file with a `.kdb` extension and up to three other files with `.sth`, `.rdb`, and `.crl` extensions.

You must save the key database password to a stash file on your computer. Save this password so that product components can use SSL without requiring any intervention from you.

The following diagram shows where these files are used in the components of i2 Analyze and the connections between them.



The environment in which you are deploying i2 Analyze might already contain files that are candidates for use as keystores or truststores. If not, you must create the required files. The files that are required in the sample scenario that is described in the instructions are summarized by component in the following list.

IBM HTTP Server

To enable SSL connection, the HTTP server requires a CMS key database (`*.kdb`). The password for this key database must be saved to a stash file. A certificate in the key database identifies the HTTP server and is used when clients connect to i2 Analyze so that they can authenticate the HTTP server. This key database is also used as a truststore to authenticate the certificate that it receives from Liberty.

The i2 Analyze client's truststore, usually located in the operating system or web browser, is used to authenticate the certificate that it receives from the HTTP server.

WebSphere Application Server Liberty

The Liberty server requires a keystore file (`*.jks`) and a truststore file (`*.jks`).

A certificate in the keystore identifies the Liberty server and is used to connect to the HTTP Server. The HTTP Server key database authenticates this certificate that it receives from Liberty.

The certificate in the Liberty truststore is used to authenticate certificates that are received from the database management system keystore and the Solr keystore.

Solr

The Solr server requires a keystore file (`*.jks`) and a truststore file (`*.jks`). Solr also requires that all Solr certificates are available in the Solr truststores and the Liberty truststore, so that individual nodes can trust one another, and Liberty can trust the Solr nodes.

The certificate in the Solr keystore identifies the Solr Server and is used to connect to Liberty.

The certificate in the Solr keystore also identifies each Solr node and is used to authenticate secure connection within Solr itself, using the Solr truststore. The certificate in the Solr truststore is used to authenticate the certificate that it received from the Solr keystore.

Database management system

To enable SSL connection, the database management system requires a keystore to connect to Liberty. The type of keystore depends on the type of database management system. For more information about SSL in your database management system, see "Configuring SSL for a DB2 instance" on page 84 or "Configure SSL for a Microsoft SQL Server" on page 86.

The certificate in the database management system keystore identifies the database management system server and is used to connect to Liberty.

In the following procedures, example commands are provided for creating the keystores, certificates, and truststores to use with each component of i2 Analyze. The instructions contain details that are based on a single-server deployment example.

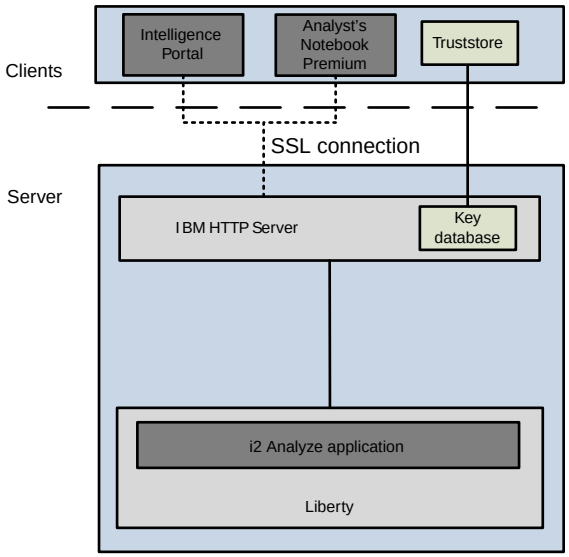
Securing the connection between clients and the HTTP server

i2 Analyze supports the use of SSL to secure the connections between the clients and the HTTP Server. After you configure the connections between the clients and the HTTP server, you can opt to configure the connection between the HTTP server and WebSphere Application Server Liberty.

About this task

To connect to i2® Analyze by using SSL, the client workstation must first trust the HTTP server because the client connects to i2 Analyze through the HTTP server. Trust is established through a certificate that is associated with the HTTP server and trusted by the client workstation.

The diagram shows the connection that you secure by completing the steps in the following procedures. It also shows the key database that is required.



You must modify the configuration of both the i2 Analyze application and the HTTP server to use SSL to secure the connection.

Creating the HTTP Server key database and certificate

The HTTP Server stores its associated certificates in a key database. You must create and populate a key database for the HTTP Server to use. In a test environment, you can create a self-signed certificate to demonstrate SSL communication.

About this task

In i2 Analyze, SSL connections that involve the HTTP server require a key database that contains a signed certificate. In a production deployment, after you create the key database, you must populate it with a certificate that is signed by a trusted certificate authority. To demonstrate a working configuration, you can create and use a self-signed certificate.

The IBM Key Management utility uses a GUI or Window Manager. If you do not have a GUI or Window Manager on your system, you can use the command line interface.

- For more information about the command line interface, see Managing keys from the command line.
- For more information about the GUI, see Managing keys with the IKEYMAN graphical interface.

Procedure

1. Create a key database.

   For example, run the following command:

   ```
   gskcapicmd -keydb -create -db "C:\IBM\i2analyze\i2-http-keystore.kdb" -pw "password" -stash
   ```

   - Save a password for the key database to a stash file by using the `-stash` attribute.
   - Set `-db` location to the directory that contains the `toolkit` directory in your deployment. For example, the file path in your deployment might be `C:\IBM\i2analyze\i2EIA`.
2. Create a self-signed certificate.

   For example, run the following command:

   ```
   gskcapicmd -cert -create -db "C:\IBM\i2analyze\i2-http-keystore.kdb" -label "httpKey" -dn
   "CN=hostname" -pw "password"
   ```

   Important: Set the value of `CN` to the fully qualified domain name for host name of the server that hosts the HTTP server. The URL that you use to connect to i2 Analyze must use the same value for the host name as the value of the `CN`. The password is the one that you saved to the stash file in step 1.
3. Extract the certificate from the key database.

   For example, run the following command:

   ```
   gskcapicmd -cert -extract -db "C:\IBM\i2analyze\i2-http-keystore.kdb" -label "httpKey" -target
   "C:\IBM\i2analyze\i2-http-certificate.cer" -pw "password"
   ```

   Set the location of the certificate to the same directory as the key database.

What to do next

To enable SSL connections to i2 Analyze, the certificate that you added to, or created in, the key database must be installed to be trusted on each client workstation.

Installing the certificate on client workstations

To enable SSL connections to i2 Analyze, the certificate in the key database must be trusted on each client workstation.

About this task

The following steps describe how to install and trust the self-signed certificate on a Windows client. To install the certificate on Linux workstations, see the documentation for your operating system.

Procedure

1. Copy the certificate to any folder on the client workstation.
2. To install the certificate, complete the following steps:
   a) Double-click the certificate file.
   b) Click **Install Certificate**, and then click **Next**.
   c) Click **Place all certificates in the following store**.
   d) Click **Browse**, select **Trusted Root Certification Authorities** and click **OK**.
   e) Click **Next**, and then click **Finish**.

      Note: If the certificate is self-signed, Windows displays a security warning because it cannot verify the self-signed certificate. Click **Yes** to accept the certificate.

Configuring the HTTP server for SSL

In a default deployment of i2 Analyze, SSL is disabled. To enable clients to connect through SSL, you must modify the configuration of both the i2 Analyze application and the HTTP server.

Before you begin

Ensure that the following attributes of the `<application>` element in the `topology.xml` are set correctly:

- The `http-server-host` attribute is set to `true`
- Ensure that the `host-name` attribute is set to the name of the machine that you are running Liberty on.

For more information about modifying the topology file, see Modifying the topology.

Procedure

1. Stop the HTTP server.

   Run the command to stop the HTTP server, `C:\IBM\HTTPserver\bin\httpd -k stop`.
2. Edit the `http-server.properties` file.
   a) Navigate to the `toolkit\configuration\environment` directory, and open the `http-server.properties` file in a text editor.
   b) Set the `http.server.ssl.enabled` property to `true`.
   c) Set the `http.server.keystore.file` property to the location of the key database file of your HTTP server.
   For example, `C:/IBM/i2analyze/i2-http-keystore.kdb`.
   d) Set the `http.server.keystore.certificate.label` property to the label of the HTTP server certificate that is in the key database file. For example, `httpKey`.
3. Open a command prompt and navigate to the `toolkit\scripts` directory.

4. To deploy i2 Analyze with the edited `http-server.properties` file, and to install the SSL configuration on the HTTP server, redeploy i2 Analyze.

   For more information, see Deploying i2 Analyze.

   The `httpd.conf` and `plugin-cfg.xml` files are updated to use the SSL configuration.
5. Restart the HTTP server.

   Run the command to start the HTTP server, `C:\IBM\HTTPserver\bin\httpd -k start`.

What to do next

If you modified the `FrontEndURI` property in your deployment, you must update it to use the HTTPS protocol. For more information about changing the `FrontEndURI` property, see "Specifying the connection URI" on page 18.

To connect to the deployment from your client, see:

- "Connecting to the Intelligence Portal from a web browser" on page 77
- "Connecting from Analyst's Notebook Premium" on page 78

Testing SSL in an i2 Analyze deployment

To ensure that a deployment allows only connections that use SSL, you can access it from a client workstation. The procedure is different depending on whether you connect through a web browser or Analyst's Notebook Premium.

Before you begin

- The certificate that the client receives from the HTTP server must be trusted on the client workstation. For more information, see "Installing the certificate on client workstations" on page 76.
- The Liberty application server must be running.

Connecting to the Intelligence Portal from a web browser

If the connection from a web browser to the HTTP server in an i2 Analyze deployment is correctly configured to use SSL, you can use the HTTPS protocol to connect to the Intelligence Portal. If the application works as you expect, then the connection is secured by SSL.

Procedure

1. Open a web browser and connect to the Onyx services at `http://host_name/apollo`.

   host_name is the fully qualified domain name or IP address of the HTTP server, and matches the Common Name value of the certificate.

   If SSL is configured correctly, you are automatically redirected to `https://host_name/apollo`.
2. Open a web browser and connect to the Opal services at `http://host_name/opal`

   host_name is the fully qualified domain name or IP address of the HTTP server, and matches the Common Name value of the certificate.

   If SSL is configured correctly, you are automatically redirected to `https://host_name/opal`.

Results

When you connect to the Intelligence Portal, the connection is secure.

Connecting from Analyst's Notebook Premium

If the connection from Analyst's Notebook Premium to the HTTP server in an i2 Analyze deployment is correctly configured to use SSL, you can use the HTTPS protocol to connect to the Onyx or the Opal services. If the application works as you expect, then the connection is secured by SSL.

Before you begin

To connect to the Onyx or the Opal services, you must install the appropriate Analyst's Notebook Premium connector for your deployment. For more information about installing the Analyst's Notebook Premium connectors, see Installing IBM i2 Analyst's Notebook Premium.

Procedure

1. Open Analyst's Notebook Premium and connect to the Onyx services at the URL `https://host_name/apollo`.

   host_name is the fully qualified domain name or IP address of the HTTP server, and matches the Common Name value of the certificate.

   Note: You cannot connect by using the HTTP protocol `http://host_name/apollo`.
2. Open Analyst's Notebook Premium and connect to the Opal services at the URL `https://host_name/opal`. *host_name* is the fully qualified domain name or IP address of the HTTP server, and matches the Common Name value of the certificate.

   Note: You cannot connect by using the HTTP protocol `http://host_name/opal`.

Results

When you connect to i2 Analyze, the connection is secure.

Securing connections with Liberty

You can use SSL to secure the connection between WebSphere Application Server Liberty and the HTTP server, the database management system, and Solr. To secure connections in i2 Analyze, the HTTP Server must trust the certificate that it receives from Liberty and Liberty must trust the certificates that it receives from the database management system and Solr.

About this task

i2 Analyze uses a Java keystore to contain the certificate that is required to identify the i2 Analyze server when it connects with the HTTP Server.

i2 Analyze uses a Java truststore to verify the certificates that are received from the database management system, and Solr.

For more information, see “SSL keystores for i2 Analyze” on page 72.

Creating the Liberty keystore and certificate

WebSphere Application Server Liberty stores certificates in Java keystore files (`.jks`). Follow the procedure to create a Java keystore, and export with the appropriate certificates.

About this task

The following steps use a self-signed certificate. In a production environment, use or request a signed certificate for Liberty from a certificate authority. Place this certificate in the Liberty keystore.

Procedure

Create a keystore and self-signed certificate for Liberty by using the Java keytool utility.

For more information, see keytool - Key and Certificate Management Tool.

a) Open a command prompt and navigate to the `i2analyze\deploy\java\bin` directory.

b) Create a keystore and certificate.

For example, run the following command:

```
keytool -genkeypair -alias "libertyKey" -keystore "C:\IBM\i2analyze\i2-liberty-keystore.jks" -
dname "CN=hostname" -keyalg RSA -storepass "password"
```

Important: Ensure that you enter values as follows:

- Enter a unique alias.
- Set the location of the keystore to the directory that contains the toolkit. In some deployments, this path might be `C:\IBM\i2EIA`.
- Set the value of `CN` to the host name of the server that hosts Liberty.
- Assign a secure password.

c) Export the certificate from the Liberty keystore.

For example, run the following command:

```
keytool -exportcert -alias "libertyKey" -keystore "C:\IBM\i2analyze\i2-liberty-keystore.jks" -
file "C:\IBM\i2analyze\i2-liberty-certificate.cer" -storepass "password"
```

What to do next

If you are using self-signed certificates, add the certificate that you exported from your Liberty keystore to the HTTP server key database. For more information, see “Adding the Liberty certificate to the HTTP key database” on page 81.

Configuring Liberty for SSL

To secure the connection between WebSphere Application Server Liberty and other components in i2 Analyze, you must configure Liberty for SSL. Update the configuration with the location of a keystore and truststore to use, and the passwords that are used to access the certificates that are contained within them.

Before you begin

Create a keystore and truststore for Liberty that contain the required certificates. For more information, see “Creating the Liberty keystore and certificate” on page 78.

About this task

Modify the i2 Analyze `topology.xml` file to specify that a secure connection must be used with the application server. Then, update the `credentials.properties` file to specify the password for the Liberty keystore and truststore files.

When the procedure is completed, it is only possible to connect to Liberty by the HTTPS protocol that uses the secure port that is defined in the port definition properties file. The non-secure port cannot be used.

Procedure

1. In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
   a) In the `<application>` element for the application server to secure, add the `secure-connection` attribute with the value of `true`.
   For example, add the attribute as highlighted in the following code:

```
<application http-server-host="true" name="opal-server"
  host-name="hostname" secure-connection="true">
```

   Note: The `host-name` attribute value must match the common name that is associated with the certificate for the application server.

   b) Add the `<key-stores>` element as a child of the `<application>` element. Then, add a child `<key-store>` element.

   For your keystore, specify the `type` as `key-store` and `file` as the full path to your keystore.

   For example, add the attribute as highlighted in the following code:

```
<application http-server-host="true" name="opal-server"
  host-name="hostname" secure-connection="true">
...
    <key-stores>
        <key-store type="key-store"
          file="C:/IBM/i2analyze/i2-liberty-keystore.jks"/>
    </key-stores>
...
</application>
```

2. Specify the keystore passwords in the credentials file.
   a) In a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
   b) Enter the password for the keystore that you specified in the `topology.xml` file.

```
ssl.keystore.password=password
```

3. Redeploy i2 Analyze so that the configuration changes are deployed to the application. For more information, see Deploying i2 Analyze.

What to do next

You must secure the connection between the HTTP Server and Liberty. For more information, see “Securing the connection between the HTTP server and Liberty” on page 80.

Securing the connection between the HTTP server and Liberty

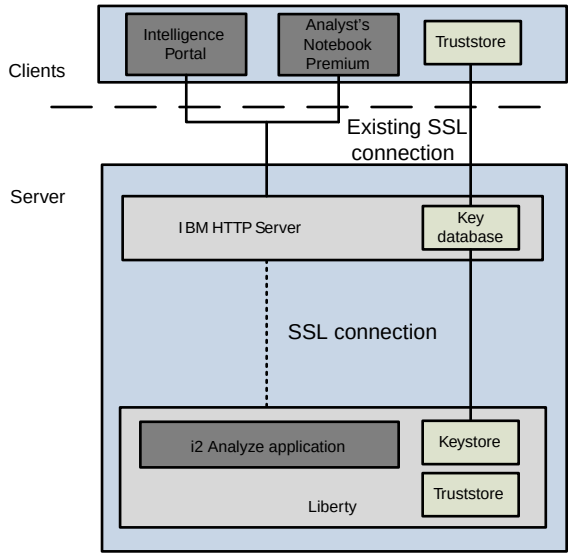
Use SSL to secure the connection between the HTTP server and WebSphere Application Server Liberty. The HTTP server key database and the Liberty truststore must contain the required certificates.

Before you begin

- Secure the connection between clients and the HTTP server. For more information, see “Securing the connection between clients and the HTTP server” on page 74.
- Configure Liberty for SSL. For more information, see “Configuring Liberty for SSL” on page 79.

About this task

The diagram shows the connection that you secure by completing the following instructions. It also shows key database and keystore that are required.



Adding the Liberty certificate to the HTTP key database

To secure the connection between the HTTP server and Liberty, you must add to the key database the certificate that is required to enable trust with Liberty.

About this task

If you are using self-signed certificates, add to the HTTP server key database the certificate that you exported from your Liberty keystore.

In a production environment, import certificates into the HTTP key database to ensure that the certificates that are received from Liberty are trusted.

Procedure

Add the certificate that you exported from the Liberty keystore into the HTTP Server key database.

For more information about exporting the certificate from the Liberty keystore, see "Creating the Liberty keystore and certificate" on page 78.

For example, run the following command:

```
gskcapicmd -cert -add -db "C:\IBM\i2analyze\i2-http-keystore.kdb" -label "libertyKey" -file
"C:\IBM\i2analyze\i2-liberty-certificate.cer" -pw "password"
```

Configuring the IBM HTTP Server

To secure the connection between the i2 Analyze application server and the HTTP server, you must arrange for the `plugin-cfg.xml` file to contain some necessary information.

Procedure

1. Stop the HTTP Server.
2. Navigate to the `toolkit\configuration\environment` directory, and open the `http-server.properties` file in a text editor.
3. Set the value of the `http.server.ssl.require.secure.backend` property to `true`, and then save and close the file.
4. Open a command prompt, and navigate to the `toolkit\scripts` directory.
5. To deploy i2 Analyze with the edited `http-server.properties` file, run the following command:

   ```
   setup -t configureHttpServer
   ```

   The `plugin-cfg.xml` file is updated to enforce that a secure connection is available between the HTTP server and Liberty.
6. Navigate to the `IBM\HTTPServer\plugins\iap\config` directory, and open the `plugin-cfg.xml` file in an XML editor.
7. In each `<ServerCluster>` element, there is a child `<Server>` element. Ensure that each of these `<Server>` elements has a child `<Transport>` element that uses the `https` protocol.

   Depending on your deployment, update the `<ServerCluster>` element with the value `"opal-server_cluster"`, or the `<ServerCluster>` element with the value `"onyx-server_cluster"` or both.

   a) Add the following code element to any of the child `<Server>` elements that do not have a child `<Transport>` element that uses the HTTPS protocol.

   ```
   <Transport Hostname="hostname" Port="portnumber" Protocol="https">
   </Transport>
   ```

   hostname is the same as the `<Transport>` element that uses the HTTP protocol and *portnumber* matches the value in the port definition properties for the application that you are securing. You can find this value in `C:\IBM\i2analyze\toolkit\configuration\environment\opal-server` or `C:\IBM\i2analyze\toolkit\configuration\environment\onyx-server`.

   b) Add the following `<Property>` elements as children of each `<Transport>` element that uses the HTTPS protocol:

   ```
   <Property Name="Keyring" Value="C:/IBM/i2analyze/i2-http-keystore.kdb"/>
   <Property Name="Stashfile" Value="C:/IBM/i2analyze/i2-http-keystore.sth"/>
   ```

   Where the `Value` attributes contain the absolute paths to the keystore for the HTTP server and the associated password stash file.

   c) Save and close the `plugin-cfg.xml` file.
8. Restart the HTTP server.

What to do next

To ensure that the configuration is correct, look in the `IBM\HTTPServer\plugins\iap\logs\plugin-cfg.log` file.

If the <Property> elements for the keyring and stashfile are not present on each <Transport> element in your plugin-cfg.xml, the following error message is displayed:

```
ERROR: ws_transport:
transportInitializeSecurity: Keyring was not set.
ERROR: ws_transport:
transportInitializeSecurity: No stashfile or keyring password given.
```

To resolve this issue, ensure that the <Property> elements for the keyring and stashfile are present on each <Transport> element in your plugin-cfg.xml.

Securing the connection between Liberty and the database

In an i2 Analyze deployment, Liberty connects to the database management system. You can secure the connection between Liberty and the database management system by using SSL.

Before you begin

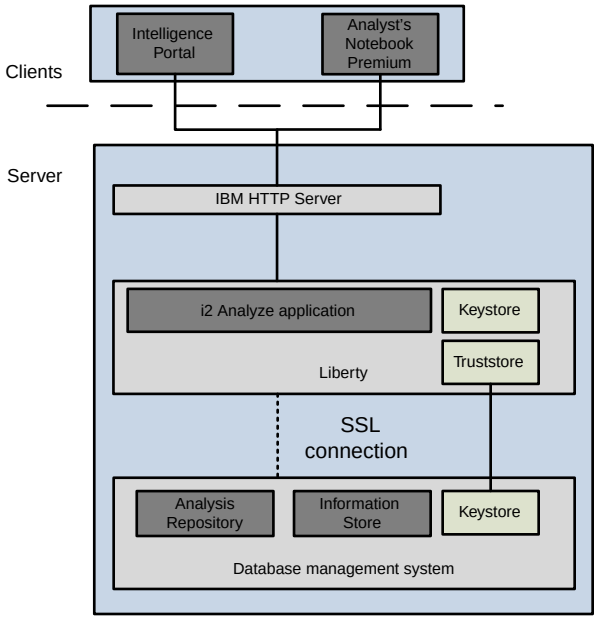
Configure Liberty for SSL. For more information, see "Configuring Liberty for SSL" on page 79.

About this task

- You can secure the connection to the database if your deployment uses the Analysis Repository on DB2 or Microsoft SQL Server, or an Information Store with the Opal services.
- You cannot use SSL to secure the connection if you are using the Analysis Repository on Oracle, or an Information Store with the Onyx services.

Setting up an SSL connection to the database is a two-part process. First, you must configure your database instance to use SSL. Then, regardless of which database management system you use, you must update the i2 Analyze configuration with the location of the truststore and the truststore password.

This diagram shows the connection that you secure by completing the steps in the following procedures. It also shows the keystore and truststore that are required.



Configuring SSL for a DB2 instance
To secure the connection between the i2 Analyze application server and the database instance, you must change the configuration of both. The DB2 Server stores its associated certificates in a key database, you must create and populate a key database for the DB2 Server to use.

About this task

In i2 Analyze, SSL connections that involve the DB2 server require a key database that contains a signed certificate. In a production deployment, after you create the key database, you must populate it with a certificate that is signed by a trusted certificate authority. To demonstrate a working configuration, you can create and use a self-signed certificate. If you are using a certificate that is signed by a certificate authority, then you can add it to the key database, you do not need to complete step 2 to create the self-signed certificate.

Ensure that you understand the details that are provided in the DB2 documentation to configure SSL for your DB2 instance. For more information, see Configuring Secure Sockets Layer (SSL) support in a DB2 instance.

For example, to run the commands in the procedure, set your Windows environment variable as follows: `set PATH=<path_to_gsk8_directory>\bin;<path_to_gsk8_directory>\lib;%PATH%`

Procedure

1. Create a key database by using the GSKCapiCmd tool.
   a) Start the GSKCapiCmd tool.

      Note: Start the GSKCapiCmd tool by using the `gskcapicmd` command. Follow the details that are provided in the DB2 documentation link for the path to the command, the required libraries, and the command syntax.

   b) Create a key database.

For example, to create the key file, run the following command:

```
gsk8capicmd_64 -keydb -create -db "C:\IBM\i2analyze\i2-db2-keystore.kdb" -pw "password" -
stash
```

Important: Ensure that you enter values as follows:

- Set the location of the key database to the directory that contains the toolkit. In some deployments, this path might be `C:\IBM\i2EIA`.
- Assign a secure password.

2. Create a self-signed certificate.

   For example, to create the self-signed certificate, run the following command:

   ```
   gsk8capicmd_64 -cert -create -db "C:\IBM\i2analyze\i2-db2-keystore.kdb" -label "dbKey" -dn
   "CN=hostname" -pw "password"
   ```

   Note: The command is a simplified version of the command in the DB2 documentation without the O, OU, L, and C values that are not required for this example. Use a `label` of `dbKey` to align with `httpKey` and `libertyKey` used in the HTTP Server and Liberty keystores. Ensure that the common name in the certificate matches the fully qualified domain name of the database instance server.

3. Extract the certificate from the key database.

   For example, to extract the certificate, run the following command:

   ```
   gsk8capicmd_64 -cert -extract -db "C:\IBM\i2analyze\i2-db2-keystore.kdb" -label "dbKey" -
   target "C:\IBM\i2analyze\i2-db2-certificate.cer" -pw "password"
   ```

4. Configure DB2 for SSL.

   For example, to configure DB2 for SSL only, run the following commands:

   ```
   db2 "UPDATE DBM CFG USING SSL_SVR_KEYDB C:\IBM\i2analyze\i2-db2-keystore.kdb"
   db2 "UPDATE DBM CFG USING SSL_SVR_STASH C:\IBM\i2analyze\i2-db2-keystore.sth"
   db2 "UPDATE DBM CFG USING SSL_SVR_LABEL dbKey"
   db2 "UPDATE DBM CFG USING SSL_SVCENAME 50001"
   db2set DB2COMM=SSL
   ```

5. Navigate to the `toolkit/scripts` directory to run setup commands.

   ```
   setup -t stop
   db2stop
   db2start
   setup -t start
   ```

What to do next

After you configure DB2, you can check the `db2diag.log` file to ensure that there are no errors with your SSL configuration.

Configure SSL for a Microsoft SQL Server
To secure the connection between the i2 Analyze application server and the database instance, you must change the configuration of both. The Microsoft SQL Server stores its associated certificates and you must create or obtain certificates for the Microsoft SQL Server to use.

Before you configure SSL for your SQL Server instance, obtain a certificate to use with it. In a demonstration system, you can create a self-signed certificate by following the instructions in *Creating and Installing SSL certificate*, in Enabling client side SSL encryption to SQL Server.

Note: Ensure that the common name in the certificate matches the fully qualified domain name of the SQL Server instance.

To configure SSL for your SQL Server instance, see Enable Encrypted Connections to the Database Engine (SQL Server Configuration Manager). Then, extract your certificate in the DER-encoded binary format as a `.cer` file. For more information, see Export a certificate.

To verify that you configured your SQL Server instance correctly, see *Test the connectivity with encryption* in Enabling client side SSL encryption to SQL Server.

Configuring i2 Analyze to connect to a database instance using SSL
To connect to a database instance by using SSL, i2 Analyze must be able to trust and verify the certificate that it receives from the database server. Those requirements mean that the certificate must also be stored in a truststore that i2 Analyze can access.

Before you begin

- Ensure that you configured Liberty for SSL. For more information, see "Configuring Liberty for SSL" on page 79.
- Your database management system must be configured for SSL. For more information, see:
  - "Configuring SSL for a DB2 instance" on page 84
  - "Configure SSL for a Microsoft SQL Server" on page 86

About this task

i2 Analyze uses a Java truststore to verify the certificate from the database server, and so you must create a truststore on your i2 Analyze server that contains the trusted certificates for your database. You can use the same truststore that you created for Liberty. For more information, see "Creating the Liberty keystore and certificate" on page 78.

For Liberty to communicate with the secured database, in the topology database element you must specify the secure connection attribute to be true and the name of the truststore that contains the database certificate. Also, specify the correct port number, which corresponds to the SSL port for the database. In the `credentials.properties` file, the correct password for the specified truststore must be added.

Procedure

1. Create the Liberty truststore and import into the truststore the certificate that you exported from the database management system.

   For example, run the following command:

   ```
   keytool -importcert -alias "dbKey" -file C:\IBM\i2analyze\i2-db2-certificate.cer -keystore
   "C:\IBM\i2analyze\i2-liberty-truststore.jks" -storepass "password"
   ```

   Enter `yes` in response to the query, `Trust this certificate?`

Important: Ensure that you enter values as follows:

- Enter a unique alias.
- Set the location of the keystore to the directory that contains the toolkit. In some deployments, this path might be `C:\IBM\i2EIA`.
- Assign a secure password.

2. Modify the i2 Analyze topology to use SSL for its database connection.
   a) In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
   b) In the `<database>` element for the database that you want to connect to with SSL, add the `secure-connection="true"` attribute.
   c) In the same `<database>` element, add the `trust-store` attribute with the location of the truststore.

      The sample scenario uses a common truststore for Liberty that you created in step1 and define in the `<key-stores>` element as shown in the example. Alternatively, you can use a specific truststore that is defined for DB2.

      ```
      <database database-type="InfoStore" dialect="db2" instance-name="DB2" database-name="ISTORE"
       edition="db2aese" xa="false" id="infostore" host-name="hostname" port-number="50001"
      version="10.5"
        secure-connection="true" trust-store="C:/IBM/i2analyze/i2-liberty-truststore.jks"/>
      ```

   d) In same element, ensure that the following attribute values are correct:
      - The `host-name` attribute value must match the common name that is associated with the certificate for the database.
      - The `port` attribute value must match value of the port number when you configured DB2 for SSL.
3. Specify the truststore password in the credentials file:
   a) In a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
   b) Enter the password for the truststore:
      - For the Analysis Repository, enter a password for the `db.write1.truststore.password` credential.
      - For the Information Store, enter a password for the `db.infostore.truststore.password` credential.

What to do next

Redeploy i2 Analyze so that the configuration changes are deployed to the application. For more information, see Deploying i2 Analyze.

Testing the deployment

To test the SSL connection between i2 Analyze and the database management system, connect to i2 Analyze. After you connect, ensure that you can create, browse, and search for data.

Procedure

1. Connect to your data store by using one of the supported clients. For more information, see Connecting clients.
2. Create, browse, and search for data to ensure that the database connection is working.

Securing the connection between Liberty and Solr

An i2 Analyze deployment with Opal services uses Apache Solr for the text indexing and search capabilities. You can secure the connection between Liberty and Solr by using SSL.

Before you begin

Configure Liberty for SSL. For more information, see “Configuring Liberty for SSL” on page 79.

Create the Liberty truststore. For more information, see “Configuring i2 Analyze to connect to a database instance using SSL” on page 86.

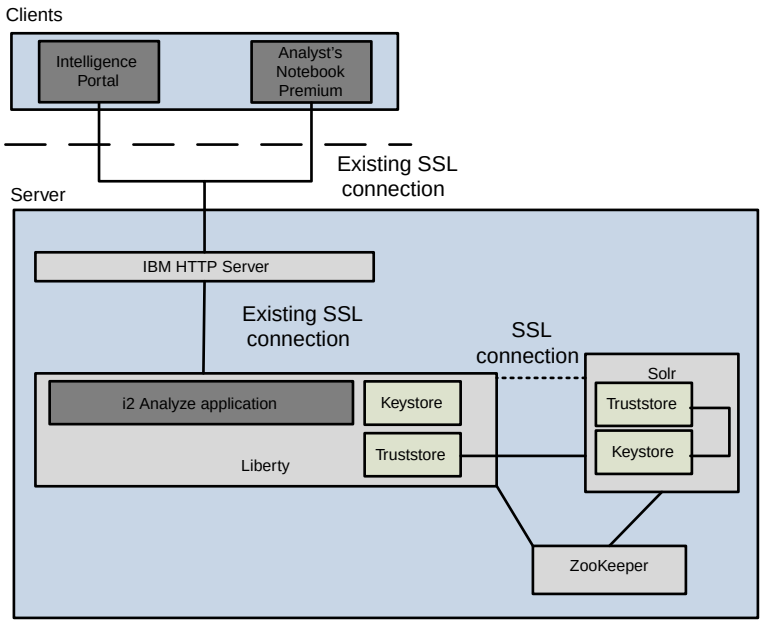
About this task

Setting up an SSL connection to the Solr is a two-part process. First, you must configure Solr to use SSL. Then, you must update the i2 Analyze configuration with the location of the Solr keystores and passwords.

The certificate in the Solr keystore is used to identify the Solr server. The certificate in the Liberty truststore is used to authenticate certificates that are received from the Solr keystore.

The certificate in the Solr keystore is also used for authentication within Solr itself, using the Solr truststore. The certificate in the Solr truststore is used to authenticate the certificate that it received from the Solr keystore.

The diagram shows the connection that you can secure by completing following the following instructions. It also includes the keystores and truststores that are required for a single server.



Note: These instructions do not cover securing the ZooKeeper connection. ZooKeeper does not support SSL encrypted communication with clients. To secure the connection to ZooKeeper, you can set up a secure connection by using an alternative method to SSL, such as an SSH tunnel.

Creating Solr keystores and certificates

Solr stores its associated certificates in keystore and truststore files (`.jks`). The certificate in the Solr keystore identifies the server that Solr is deployed on. This certificate is checked against the certificate in the Liberty truststore when Liberty attempts to connect to Solr.

About this task

To ensure that communication between the i2 Analyze application server and the Solr index is secured, create a keystore and truststore for Solr.

The following steps use a self-signed certificate. In a production environment, use or request a signed certificate for Liberty from a certificate authority. Place this certificate in the Liberty keystore.

For Solr to work correctly, there must be a keystore for each server on which Solr is deployed. Each keystore contains a certificate that identifies the server. Solr also requires that all Solr certificates are available in the Solr truststore and the Liberty truststore, so that individual nodes can trust one another, and Liberty can trust the Solr nodes.

Depending on the topology of your deployment, you might need to create a separate keystore and truststore for each Solr node.

Procedure

1. Create a keystore and export a certificate for Solr by using the Java keytool utility.

   For more information, see keytool - Key and Certificate Management Tool.

   a) Create a keystore and a certificate.

   For example, run the following command:

   ```
   keytool -genkeypair -alias "solrKey" -keystore "C:\IBM\i2analyze\i2-solr-keystore.jks" -
   dname "CN=hostname" -keyalg RSA -storepass "password"
   ```

   Important: Ensure that you enter values as follows:

   - Set the value of `CN` to the host name of the server that hosts Solr.
   - Set the location of the key database to the directory that contains the toolkit. In some deployments, this path might be `C:\IBM\i2EIA`.
   - Set the `storepass` attribute to be a secure password for the keystore.

   Enter the key password for the Solar key or press the Return key to confirm that the key password is the same as keystore password.

   b) Export the certificate from the Solr keystore.

   For example, run the following command.

   ```
   keytool -exportcert -alias "solrKey" -keystore "C:\IBM\i2analyze\i2-solr-keystore.jks" -
   file "C:\IBM\i2analyze\i2-solr-certificate.cer" -storepass "password"
   ```

   Enter the password that you set for the keystore in the previous step.

2. Create the Solr truststore and import the required certificates.

   If you are using a self-signed certificate, import the certificate that you exported from the Solr keystore in step 1b.

For example, run the following command:

```
keytool -importcert -alias "solrKey" -keystore "C:\IBM\i2analyze\i2-solr-truststore.jks" -file
"C:\IBM\i2analyze\i2-solr-certificate.cer" -storepass "password"
```

In response to the prompt, enter `yes` to trust the certificate.

Configuring an SSL connection to Solr

To secure the connection between the i2 Analyze application server and Solr, you must change the configuration of both. i2 Analyze manages the configuration of Solr, the i2 Analyze configuration must define the keystore and truststore for Solr.

Before you begin

- Ensure that you configured Liberty for SSL. For more information, see "Configuring Liberty for SSL" on page 79.
- You must have the appropriate keystore set up for your Solr deployment. For more information, see "Creating Solr keystores and certificates" on page 89.

About this task

i2 Analyze uses a Java truststore to verify the certificate from the Solr server, and so you must create or reuse a truststore on your i2 Analyze server to contain the trusted certificates for your Solr system. You can use the same truststore that you created for Liberty to hold the certificate from the database management system. For more information, see "Configuring i2 Analyze to connect to a database instance using SSL" on page 86.

Procedure

1. Ensure that the Liberty instance is stopped before you implement the configuration changes, by running the command: `setup -t stop`

   Ensure that the Liberty instance is stopped, otherwise you encounter an error if you try to run the command when you complete the configuration changes.
2. Create the truststore and import the certificate that you exported from the Solr keystore into the truststore.

   This truststore can be the truststore that you created for the certificate that you exported from the database management system. Alternatively, you create a new truststore if the file does not exist.

   For example, run the following command:

   ```
   keytool -importcert -alias "solrKey" -keystore "C:\IBM\i2analyze\i2-liberty-truststore.jks" -
   file "C:\IBM\i2analyze\i2-solr-certificate.cer" -storepass "password"
   ```

   In response to the prompt, enter `yes` to trust the certificate.

   Important: Ensure that you enter values as follows:

   - Enter a unique alias.
   - Set the location of the keystore to the directory that contains the toolkit. In some deployments, this path might be `C:\IBM\i2EIA`.
   - Assign a secure password.
3. Modify the `topology.xml` file to specify SSL for its Solr connection.

a) In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.

b) In the `<solr-cluster>` element for the Solr cluster that you want to connect to with SSL, add the `secure-connection` attribute with the value of `true`.
For example, add the attribute as highlighted in the following code:

```
<solr-cluster id="is_cluster" zookeeper-id="zoo" secure-connection="true">
```

c) Add the `key-store` and `trust-store` attributes to either the `<solr-cluster>` or to the `<solr-node>` element.

Add the attribute values as defined:

key-store

The path to the Solr keystore. For more information, see "Creating Solr keystores and certificates" on page 89. Reference step 1a.

trust-store

The path to the Solr truststore. For more information, see "Creating Solr keystores and certificates" on page 89. Reference step 2a.

- For example, add the attribute as highlighted in the `<solr-clusters>` element:

```
<solr-cluster id="is_cluster" zookeeper-id="zoo" secure-connection="true"
  key-store="C:\IBM\i2analyze\i2-solr-keystore.jks" trust-store="C:\IBM\i2analyze\i2-solr-
truststore.jks">
```

- For example, add the attribute as highlighted in the `<solr-node>` element:

```
<solr-node memory="2g" data-dir="C:\IBM\i2analyze\data\solr" host-name="hostname"
id="node1" port-number="8983"
  key-store="C:\IBM\i2analyze\i2-solr-keystore.jks" trust-store="C:IBM\i2analyze\i2-solr-
truststore.jks">
```

Note: The `host-name` attribute value must match the common name that is associated with the certificate for Solr. For more information, see "Creating Solr keystores and certificates" on page 89. Reference step 1a.

4. Modify the `topology.xml` file to add the Liberty truststore.

Add a child `<key-store>` element. For your keystore, specify the `type` as `trust-store` and `file` as the full path to your truststore.

For example, add the element as highlighted in the following code:

```
<application http-server-host="true" name="opal-server"
  host-name="hostname" secure-connection="true">
...
    <key-stores>
        <key-store type="key-store"
          file="C:\IBM\i2analyze\i2-liberty-keystore.jks"/>
        <key-store type="trust-store"
          file="C:\IBM\i2analyze\i2-liberty-truststore.jks"/>
    </key-stores>
...
</application>
```

5. Specify the truststore and keystore passwords in the credentials file.
   a) In a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
   b) Enter the passwords for the Solr keystore and truststore that you specified in the topology file.

```
solr.truststore.password=password
solr.keystore.password=password
```

   c) Enter the password for the Liberty truststore that you specified in the topology file.

```
ssl.truststore.password=password
```

What to do next

Redeploy i2 Analyze so that the configuration changes are implemented in i2 Analyze and Solr configuration. For more information, see Deploying i2 Analyze.

Connect to your Information Store. For more information, see Connecting IBM i2 Analyst's Notebook Premium to IBM i2 Analyze.

Search for data to ensure that the Solr connection is working.

Resources for system protection

In order to protect your system from external forces, you must implement system controls that prevent or mitigate the effect of attacks. Although IBM does not manage login configuration, and the responsibility for protection of your network from external attack remains yours, the following communities provide a starting point for your investigation into preventative methods.

The Open Web Application Security Project

The Open Web Application Security Project Foundation is a not-for-profit organization that is dedicated to enabling organizations to conceive, develop, operate and maintain applications that can be trusted. In particular, see https://www.owasp.org/index.php/Blocking_Brute_Force_Attacks and https://www.owasp.org/index.php/Authentication_Cheat_Sheet#Password_Complexity.

SANS Institute

The System-Admin, Audit, Network and Security Institute is the largest source for information security training and security certification in the world. It also develops, maintains, and makes available at no cost, the largest collection of research documents about various aspects of information security, and it operates the Internet's early warning system - the Internet Storm Center. In particular, see https://www.sans.org/security-resources/policies/general/pdf/password-construction-guidelines

Common Weakness Enumeration

CWE™ is a community-developed list of common software security weaknesses. It serves as a common language, a measuring stick for software security tools, and as a baseline for weakness identification, mitigation, and prevention efforts.

In particular, see http://cwe.mitre.org/top25/index.html#CWE-307.

Configuring SPNEGO single sign-on for i2 Analyze

The following section describes how to configure IBM i2 Analyze with Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) single sign-on. The instructions detail how to configure SPNEGO single sign-on with an existing deployment of i2 Analyze.

There are many different single sign-on technologies. This section defines a SPNEGO single sign-on setup with workstations that are members of the same Microsoft Active Directory domain. i2 Analyze uses the users and groups in Active Directory to determine the authorization of users.

The instructions assume that the following prerequisites are installed and accessible:

- A Microsoft Windows® Server running an Active Directory Domain Controller and associated Kerberos Key Distribution Center (KDC).
- A Microsoft Windows® domain member (client) with a web browser that supports the SPNEGO authentication mechanism.
- A working deployment of i2 Analyze that can be accessed by users in Active Directory.

For information on the prerequisites that are required, see the Before you begin section of Configuring SPNEGO authentication in Liberty in IBM Knowledge Center.

Attention: IBM takes reasonable steps to verify the suitability of i2 Analyze for Internet deployment. However, it does not address lower-level issues such as guarding networks against penetration, securing accounts, protecting against brute force attacks, configuring firewalls to avoid DoS or DDoS attacks, and the like. For your deployment of i2 Analyze, follow industry-standard practices and recommendations for protection of your systems. IBM accepts no liability for the consequences of such attacks on your systems. This information is not intended to provide instructions for managing key databases or certificates.

Intended audience

This section is intended for readers who are familiar with configuring and managing domain controllers, Microsoft Active Directory, and have an understanding of SPNEGO single sign-on.

i2 Analyze with SPNEGO single sign-on

By following the IBM i2 Analyze Deployment Guide, i2 Analyze is configured to use user names and passwords that are stored in a file-based registry. By configuring the deployment to use SPNEGO single sign-on, a user is logged in through the domain client workstation that they are logged in to.

After a user logs in to a single sign-on environment, they are authenticated with any systems that they have access to. i2 Analyze can be configured to allow authentication through SPNEGO single sign-on, with authorization through Active Directory.

SPNEGO single sign-on enables users to log in to a Microsoft domain controller, and be authenticated within the single sign-on environment. In SPNEGO single sign-on, to change the user that is logged in to i2 Analyze, the user must log out of the workstation, and a new user must log in to the workstation.

SPNEGO single sign-on planning

Planning an implementation of SPNEGO single sign-on with i2 Analyze ensures that the system you create matches the needs of your organization. It is important to understand the organizational requirements and your environment before you begin to plan.

Before you start implementing the solution, ensure that you understand the following aspects of the proposed system:

- What SPNEGO single sign-on is, and the implications of implementing it in your environment. For more information, see Single sign-on for HTTP requests using SPNEGO web authentication in IBM Knowledge Center.
- The physical architecture that is required to use SPNEGO single sign-on.
- The value of the `UserGroup` attribute of each `<GroupPermissions>` element to use with your i2 Analyze deployment. For more information, see the i2 Analyze Security White Paper.

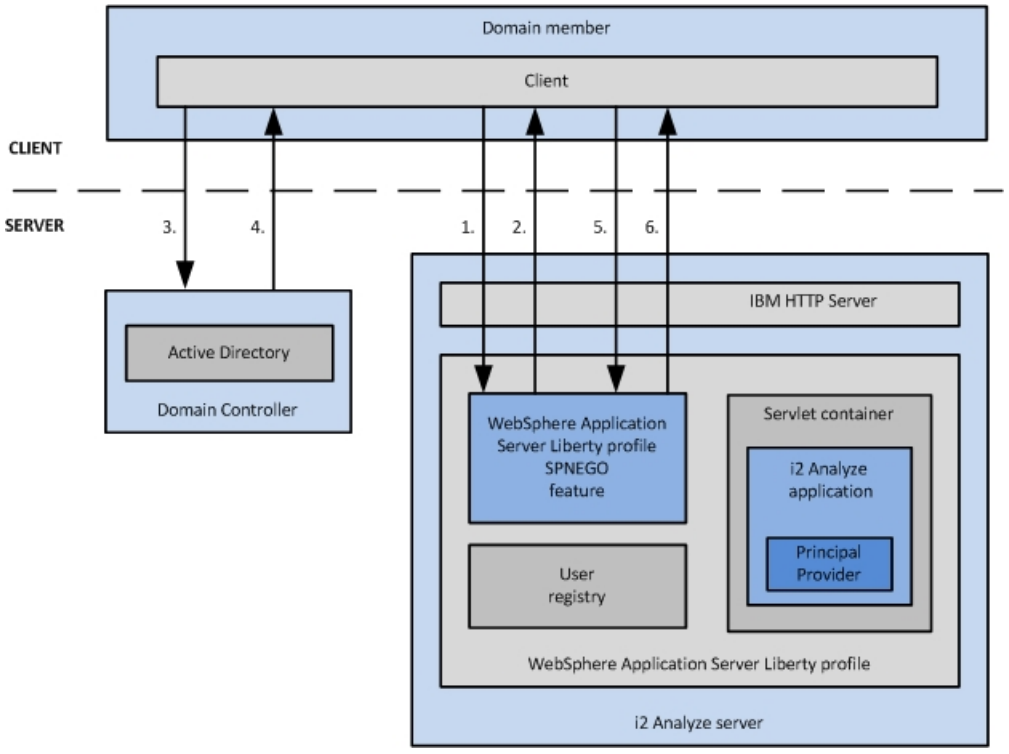
i2 Analyze with SPNEGO single sign-on model

Configuring i2 Analyze to use SPNEGO single sign-on changes the way that users authenticate with the platform. A deployment that uses SPNEGO single sign-on requires the user to access i2 Analyze on a workstation that is a member of the same domain as i2 Analyze.

Authentication

When i2 Analyze is configured to use SPNEGO single sign-on, the authentication sequence between the client and the platform matches the following steps and the associated diagram:

1. The client attempts to connect to WebSphere Application Server Liberty profile with an `HTTP/Post/Get` request.
2. WebSphere Application Server Liberty profile returns `HTTP 401` with a `Negotiate` header.
3. The client requests a SPNEGO token from the domain controller.
4. The domain controller returns a SPNEGO token to the client.
5. The client attempts to connect to WebSphere Application Server Liberty profile with an `HTTP/Post/Get` request and the SPNEGO token.
6. On successful authentication, the client receives a Lightweight Third-Party Authentication (LTPA) token in a cookie. During normal operation, the client passes the cookie back to i2 Analyze.

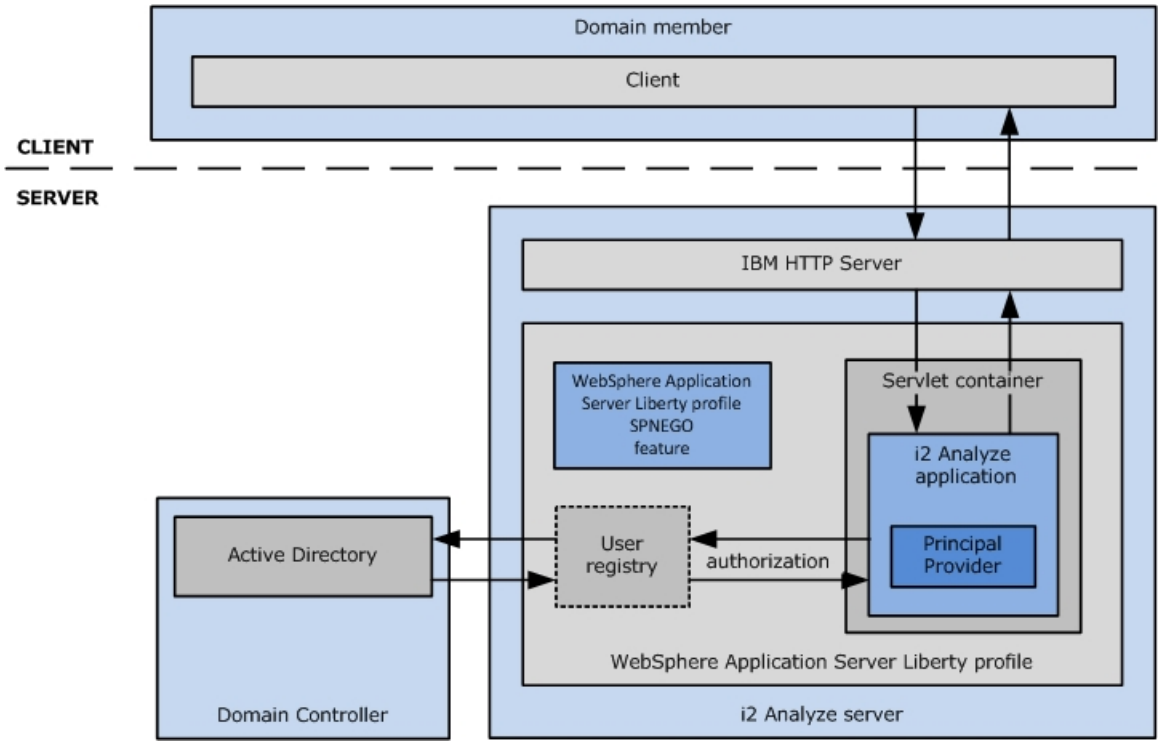


Authorization

After the user is authenticated, they are logged in to i2 Analyze. To define the data that the user has access to, the user must be authorized by i2 Analyze.

For authorization, the i2 Analyze application communicates with Active Directory, through the WebSphere Application Server Liberty profile user registry APIs to retrieve information about the current user. The principal provider then maps the retrieved information to security dimension values in the i2 Analyze security schema.

The following diagram shows how authorization works in i2 Analyze:



Implementing SPNEGO single sign-on

SPNEGO single sign-on enables users to log in to a domain client workstation, and be authenticated with i2 Analyze. Complete any configuration changes to i2 Analyze, and test the system.

Before you begin

- For information on the prerequisites that are required, see the Before you begin section of Configuring SPNEGO authentication in Liberty.
- If you previously deployed i2 Analyze with basic authentication, you must remove or comment out the complete `<basicRegistry>` element from your user registry.

About this task

Populate Active Directory with the correct users and groups for your environment. Then, complete the configuration steps for WebSphere Application Server Liberty profile and the i2 Analyze application. Redeploy i2 Analyze, and connect to i2 Analyze from a client workstation.

Configuring Microsoft Active Directory

The users that are in Microsoft Active Directory are used to authenticate with i2 Analyze. The groups that are in Active Directory are used for authorization in i2 Analyze.

About this task

Create Microsoft Active Directory groups that match the value of the `UserGroup` attribute of each `<GroupPermissions>` element in the i2 Analyze security schema file.

The groups that you create in Microsoft Active Directory are used for authorization in i2 Analyze. To identify the groups correctly, you must ensure that the names of groups in Active Directory exactly match the value of the `UserGroup` attribute of each `<GroupPermissions>` element in the security schema.

Note: The security schema that the deployment uses is defined in the `ApolloServerSettingsMandatory.properties` file. The security schema, and properties files are in the `toolkit\configuration\fragments\common\WEB-INF\classes` directory.

In a single sign-on setup, the following users must be present in Active Directory:

- A user for the server that hosts the i2 Analyze application, that is mapped to a Kerberos Service Principal Name (SPN).
- The users that are used to log in to i2 Analyze.

To authorize users, the following groups must be present in Active Directory:

- A group for each of the group permission elements in the i2 Analyze security schema.
- A group for administrators.

Procedure

1. Create the Microsoft Active Directory groups.

   For more information, see How to Create a Group in Active Directory.

   a) Open the Microsoft Active Directory groups controller.

   b) Create groups whose names exactly match the value of the `UserGroup` attribute of each `<GroupPermissions>` element in the i2 Analyze security schema file.
2. Create any Microsoft Active Directory users.

   Create user accounts that can be used to log in to i2 Analyze.

   For more information, see How to Create a Domain Account in Active Directory.
3. Make each user a member of the correct groups for your environment.

   The groups that the user is a member of in Active Directory are used for authorization in i2 Analyze.

   For more information, see Adding Users to an Active Directory Group.

Results

The users that can access i2 Analyze are created, and are members of the groups that define their access levels.

Configuring WebSphere Application Server Liberty profile

Create the Kerberos service principal name (SPN) and keytab file for the server that hosts WebSphere Application Server Liberty profile that runs the i2 Analyze application. Edit the WebSphere Application Server Liberty profile configuration to use SPNEGO single sign-on and the Active Directory registry.

Before you begin

To use SPNEGO single sign-on with the Liberty server that is deployed by i2 Analyze and other WebSphere Application Server servers, each server must use the same LTPA keys file. For more information about LTPA, see Authentication - LTPA.

The value that is set for the `ltpakeys.password` property in the `credentials.properties` file must match the password that is required to import the keys from the LTPA keys file. If you change the password in the `credentials.properties` file, you must redeploy i2 Analyze for the password change to take effect.

Procedure

1. Configure WebSphere Application Server Liberty profile to use SPNEGO single sign-on by using the first two steps in Configuring SPNEGO authentication in Liberty as a reference.
   a) Create the Kerberos SPN and keytab files on the domain controller.

      Note: Ensure that the host file on the Active Directory server uses the full host name, including the domain name, for the i2 Analyze server. Remove any entries that use only the short name for the i2 Analyze server. The value in the host file must match the value that is used for the SPN.

   b) Configure the server that hosts WebSphere Application Server Liberty profile, and WebSphere Application Server Liberty profile.
2. Configure WebSphere Application Server Liberty profile to use the Microsoft Active Directory registry by using the instructions in Configuring LDAP user registries with Liberty as a reference.
   a) Complete step 1 to add the features to the `server.xml` file.
   b) Complete step 4 by using the Microsoft Active Directory Server example to populate the `<ldapRegistry>` element.

      Note: This information does not cover the configuration of Secure Sockets Layer (SSL) between WebSphere Application Server Liberty profile and Active Directory. Do not include the `<ssl>` and `<keyStore>` elements from the example, in your `server.xml`.

   c) Ensure that the mapping between Active Directory and the i2 Analyze security schema is correct. Add the following code after the `<ldapRegistry>` element in the `server.xml` file:

      ```
      <federatedRepository>
        <primaryRealm name="">
          <participatingBaseEntry name=""/>
          <groupSecurityNameMapping inputProperty="cn" outputProperty="cn"/>
        </primaryRealm>
      </federatedRepository>
      ```

      Populate the empty `name` attribute values by using the following information:

      - The `<primaryRealm>` element's `name` attribute has the same value as the `realm` attribute of the `<ldapRegistry>` element.
      - The `<participatingBaseEntry>` element's `name` attribute has the same value as the `baseDN` attribute as the `<ldapRegistry>` element.

By default, all requests to access protected resources use SPNEGO authentication. If you previously deployed i2 Analyze with basic authentication, you must ensure that the basic registry is not present in the `user.registry.xml` file.

3. Using an XML editor, either remove or comment out the complete `<basicRegistry>` element in the `i2analyze\deploy\wlp\usr\shared\config\user.registry.xml` file.

Configuring the i2 Analyze application

Update the `web.xml` file that is used in the `onyx-services-ar` application to enable SPNEGO authentication. Then, set the security administrator group to the value used in Microsoft Active Directory.

About this task

Use the following steps to update the `web.xml` that applies to the `onyx-services-ar` application server.

Procedure

1. Edit the `web.xml` file.
   a) Using a text editor, open the `toolkit\configuration\fragments\onyx-services-ar\WEB-INF\web.xml` file.
   b) Comment out the following lines from the `web.xml` file:

```
<filter>
    <filter-name>SSOAuthenticationFilter</filter-name>
    <description>
        Only for use in SSO. Must be removed if using
        SPNEGO Kerberos authentication.
    </description>
    <filter-class>
         com.i2group.apollo.servlet.SSOAuthenticationFilter
    </filter-class>
</filter>

<filter-mapping>
    <filter-name>SSOAuthenticationFilter</filter-name>
    <url-pattern>/services/InfoService</url-pattern>
</filter-mapping>
```

2. Using a text editor, open `toolkit\configuration\environment\`*`server`*`\environment-advanced.properties`. Edit the value of the `security.administrator.group` property to the name of the group in Active Directory to use for administrators. For example, `i2Admins`.

   The mapping to the administrator group must be in terms of the Common Name that is defined in the `<participatingBaseEntry>` element of the `<federatedRepository>` element in the `server.xml` file. For more information about the `<federatedRepository>` element, see "Configuring WebSphere Application Server Liberty profile" on page 97.

Redeploying i2 Analyze

Any time that you change the configuration of i2 Analyze, you must redeploy the system.

Procedure

1. Open a command prompt and navigate to `toolkit\scripts`.

2. Ensure that the application servers are stopped. To stop the application server, run the following command:

```
setup -t stop
```

3. To deploy i2 Analyze components, run the following command:

```
setup -t deploy
```

   Use the information in Troubleshooting the deployment process to ensure that i2 Analyze is deployed successfully.

4. To start the application server, open a command prompt and navigate to `toolkit\scripts`. Then, run the following command:

```
setup -t start
```

5. Start, or restart, the HTTP server that hosts the reverse proxy.

Testing i2 Analyze with SPNEGO single sign-on on the client workstation

To test the SPNEGO single sign-on setup, connect to i2 Analyze from a domain client workstation. The user that is logged in to the domain client is logged in to i2 Analyze.

Before you begin

- Your web browser must be configured according to step 3 in Configuring SPNEGO authentication in Liberty.
- You must be logged in to the client workstation as one of the users in the domain controller, who is in at least one group per security dimension in the i2 Analyze security schema.

About this task

Log in to the client workstation as users with different access levels. For each user, complete the following steps to demonstrate that authorization is working correctly when you are using SPNEGO single sign-on.

Procedure

1. Open your web browser, and navigate to `http://`*`host_name`*`/apollo` (where *host_name* is the fully qualified domain name or IP address of the server that hosts the i2 Analyze application).
2. Create, browse, and search for data, to ensure that your are authenticated and authorized with the platform correctly.

   Note: When you create items, ensure that the permissions are set up so that you have access to view them.

Resources for system protection

In order to protect your system from external forces, you must implement system controls that prevent or mitigate the effect of attacks. Although IBM does not manage login configuration, and the responsibility for protection of your network from external attack remains yours, the following communities provide a starting point for your investigation into preventative methods.

The Open Web Application Security Project

The Open Web Application Security Project Foundation is a not-for-profit organization that is dedicated to enabling organizations to conceive, develop, operate and maintain applications that

can be trusted. In particular, see https://www.owasp.org/index.php/Blocking_Brute_Force_Attacks and https://www.owasp.org/index.php/Authentication_Cheat_Sheet#Password_Complexity.

SANS Institute

The System-Admin, Audit, Network and Security Institute is the largest source for information security training and security certification in the world. It also develops, maintains, and makes available at no cost, the largest collection of research documents about various aspects of information security, and it operates the Internet's early warning system - the Internet Storm Center. In particular, see https://www.sans.org/security-resources/policies/general/pdf/password-construction-guidelines

Common Weakness Enumeration

CWE™ is a community-developed list of common software security weaknesses. It serves as a common language, a measuring stick for software security tools, and as a baseline for weakness identification, mitigation, and prevention efforts.

In particular, see http://cwe.mitre.org/top25/index.html#CWE-307.

Configuring X.509 client certificate authentication with i2 Analyze

You can enable your deployment to use X.509 client certificate authentication. After successful configuration, users can log in to i2 Analyze with client certificates instead of user names and passwords.

After you configure client certificate authentication, a user does not need to enter a user name and password separately. Each certificate is associated with a single user in the user registry to enable authentication. Anyone that has access to a client certificate can log in to i2 Analyze as the user associated with that certificate without entering a password.

To enable a user to log in using a client certificate, the client certificate must be installed in the user's personal certificate store on the workstation they are using to access i2 Analyze. After the client certificate is installed in the personal certificate store, the user can use the certificate to log in to i2 Analyze through the Intelligence Portal or Analyst's Notebook Premium.

When a user connects to the Intelligence Portal, the web browser prompts the user to choose the certificate to use to log in to i2 Analyze. If the user is using Analyst's Notebook Premium, the user chooses the certificate to use to log in when they open a connection to the Analysis Repository or Information Store.

Intended audience

This information is intended for readers who are familiar with managing key databases and certificates, user authentication mechanisms, and the i2 Analyze toolkit.

Prerequisites

The starting point for configuring client certificate authentication is a deployment of i2 Analyze that is configured to use Secure Sockets Layer on connections to the HTTP Server, and between the HTTP Server and Liberty. For more information about configuring Secure Sockets Layer on connections to the HTTP Server, see Configuring Secure Sockets Layer with i2 Analyze.

Attention: IBM takes reasonable steps to verify the suitability of i2 Analyze for Internet deployment. However, it does not address lower-level issues such as guarding networks against penetration, securing accounts, protecting against brute force attacks, configuring firewalls to avoid DoS or DDoS attacks, and the like. For your deployment of i2 Analyze, follow industry-standard practices and recommendations for protection of your systems. IBM

accepts no liability for the consequences of such attacks on your systems. This information is not intended to provide instructions for managing key databases or certificates.

Client certificates

The client certificates that are used to authenticate users must be signed by a certificate authority that is trusted by the i2 Analyze server.

The common name in a client certificate must match a user name in the i2 Analyze user registry. A user that selects such a certificate logs in to i2 Analyze as the corresponding i2 Analyze user.

You can have as many client certificates as you require. Each certificate is associated with a single user in the user registry. Each certificate can be installed on any number of workstations. Each workstation can have any number of certificates installed.

To demonstrate a working configuration, you can use a self-signed client certificate. For more information, see “Creating a self-signed client certificate” on page 101. However, in a production deployment you must use certificates that are signed by a certificate authority that is trusted by the i2 Analyze server.

There are many methods for obtaining an X.509 certificate that is signed by a certificate authority. When you receive a signed certificate, you also receive signer certificates so that you can trust the client certificates that are signed by that certificate authority. If the certificate authority that signed your certificates is not already trusted within the key database, you must add any signer certificates to the key database so that the certificate authority is trusted.

For information about managing key databases, certificates, and trusted certificate authorities using the IBM Key Management utility, see Managing keys with the IKEYMAN graphical interface.

Creating a self-signed client certificate

The client certificate is used to log in and authenticate a user with i2 Analyze. Use the IBM Key Management utility to create a self-signed certificate.

About this task

Create a self-signed certificate to use as a client certificate to demonstrate a working configuration. If you are using client certificates that are signed by a certificate authority, you do not need to complete the following instructions.

Procedure

1. Start the IBM Key Management utility.

   For more information, see Starting the Key Management utility user interface.

   Note: The IBM Key Management utility uses a GUI or Window Manager. If you do not have a GUI or Window Manager on your system, you can use the command line interface to complete the same actions. For more information about the command line interface, see Key Management utility command-line interface (gskcmd) syntax.
2. Open the key database that is used for Secure Sockets Layer (SSL) connections. If you followed the instructions to set up the SSL example, the key database file is `IBM\i2analyze\i2-http-keystore.kdb`.

   For more information about opening a key database, see Working with key databases.
3. Create a self-signed certificate.

   For more information, see Creating a self-signed certificate.

   a) Set the **Key Label** to a value that enables you to identify the user.
   For example, `Jenny`.

b) Ensure that the value of **Common Name** matches the name of a user in the user registry for i2 Analyze.

If you are using the example user registry, set the value of **Common Name** to `Jenny`.

Note: The user name cannot contain a comma (,).

For this example, you can use the default values for the remaining properties.

4. Export the certificate and private key from the key database.
   a) Click **Export/Import**.
   b) Ensure that **Export Key** is selected.
   c) From the **Key file type** list, select PKCS12.
   d) Set the **File name** to a value that enables you to identify the user.
   For example, `Jenny.p12`.
   e) Ensure that **Location** is set to the same directory as the key database.
   f) Click **OK**.
   g) When you are prompted, provide a password that is used to access the keys.
5. Extract the certificate from the key database.
   a) Click **Extract Certificate**.
   b) From the **Data type** list, select Binary DER data.
   c) Set the **Certificate file name** to a value that enables you to identify the user.
   For example, `Jenny.der`.
   d) Ensure that **Location** is set to the same directory as the key database.
   e) Click **OK**.

Configuring the key databases

To enable the i2 Analyze server to trust the client certificates, you must ensure that the signer of your client certificates is trusted within the i2 Analyze key database. You must also create a copy of the keystore as a truststore that WebSphere Application Server Liberty uses.

About this task

If you are using client certificates that are signed by a certificate authority, ensure that the certificate authority that signed the certificates is trusted within the i2 Analyze key database.

You can list the certificate authorities that are trusted within a key database in the IBM Key Management utility. For more information, see Listing certificate authorities.

After you add the certificate to the key database, create a truststore that WebSphere Application Server Liberty uses. The truststore is a copy of the i2 Analyze key database.

Procedure

1. Start the IBM Key Management utility.

   For more information, see Starting the Key Management utility user interface.

   Note: The IBM Key Management utility uses a GUI or Window Manager. If you do not have a GUI or Window Manager on your system, you can use the command line interface to complete the same actions. For more information about the command line interface, see Key Management utility command-line interface (gskcmd) syntax.

2. Open the key database that is used for Secure Sockets Layer (SSL) connections. If you followed the instructions to set up the SSL example, the key database file is `IBM\i2analyze\i2-http-keystore.kdb`.

   For more information about opening a key database, see Working with key databases.
3. Add the certificates to the key database, to ensure that the certificates received from the client are trusted.
   a) In the IBM Key Management utility, with the i2 Analyze key database open, select **Signer Certificates** from the list in the **Key database content** pane.
   b) Click **Add**.
   c) Click **Browse**, and locate your certificate.

   Note: When you are using a self-signed client certificate, add the self-signed client certificate as a signer certificate. For example, `Jenny.der`.
4. The Liberty truststore must contain the certificates to ensure that the certificates received from the client are trusted.
   a) Run the following command to import the required certificate into the truststore. If the truststore does not exist, it is created.:

   ```
   keytool -importcert -alias "signerKey" -keystore "C:\IBM\i2analyze\i2-liberty-truststore.jks" -file "C:\IBM\i2analyze\signer-certificate.cer" -storepass "password"
   ```

   Note: When you are using a self-signed client certificate, add the self-signed client certificate as a signer certificate. For example, `Jenny.der`.

Results
The key database contains the signer certificates so that the client certificates can be trusted. The truststore is populated so that Liberty can use it to trust the client certificates.

Configuring i2 Analyze

To enable a user to log in using a client certificate, you must modify some of the configuration files for i2 Analyze.

About this task
Add a rewrite rule that enables client authentication on the IBM HTTP Server to the i2 Analyze configuration. Then, update the `web.xml` file for the application to enable client certificate authentication.

Procedure

1. Using a text editor, open the `configuration\environment\proxy\http-custom-rewrite-rules.txt` file. Add the following line between the `!Start_After_Rules!` and `!End_After_Rules!` lines to enable client certificate authentication:

   ```
   SSLClientAuth Optional
   ```
2. In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
   a) Add a child `<key-store>` element to the `<key-stores>` element.

   For your truststore, specify the `type` as `trust-store` and `file` as the full path to your truststore.

For example, add the attribute as highlighted in the following code:

```
<application http-server-host="true" name="opal-server"
  host-name="hostname" secure-connection="true">
...
    <key-stores>
        <key-store type="key-store"
          file="C:\IBM\i2analyze\i2-liberty-keystore.jks"/>
        <key-store type="trust-store"
          file="C:\IBM\i2analyze\i2-liberty-truststore.jks"/>
    </key-stores>
...
</application>
```

b) Specify the truststore password in the credentials file. In a text editor, open the `toolkit\configuration\environment\credentials.properties` file and enter a password for the truststore that you specified in the `topology.xml` file.

```
ssl.truststore.password=password
```

3. Using an XML editor, open the `wlp\usr\servers\server\server.xml` file. Modify the `<ssl>` element with the id `defaultSSLConfig` to include `clientAuthenticationSupported="true"`. For example:

```
<ssl clientAuthenticationSupported="true"
  id="defaultSSLConfig"
  keyStoreRef="defaultKeyStore"
  trustStoreRef="defaultTrustStore"/>
```

4. Edit the `web.xml` files that are associated with your deployment.

a) If you are configuring the Opal services, use an XML editor to modify the `toolkit\configuration\fragments\opal-services-is\WEB-INF\web.xml` file.

Comment out the following lines so that form based authentication is not used:

```
<login-config>
    <auth-method>FORM</auth-method>
    <realm-name>Form-Based Authentication Area</realm-name>
    <form-login-config>
        <form-login-page>/login.html</form-login-page>
        <form-error-page>/login.html?failed</form-error-page>
    </form-login-config>
</login-config>
```

In the login configuration section, add the following lines to define the client certificate authentication method:

```
<login-config>
    <auth-method>CLIENT-CERT</auth-method>
    <realm-name>WebRealm</realm-name>
</login-config>
```

b) If you are configuring the Onyx services, use an XML editor to modify the `toolkit\configuration\fragments\onyx-services-ar\WEB-INF\web.xml` file.

Comment out the following lines so that the authentication header is not filtered:

```
<filter>
    <filter-name>SSOAuthenticationFilter</filter-name>
    <description>
        Only for use in SSO. Must be removed if using
        SPNEGO Kerberos authentication.
    </description>
    <filter-class>
        com.i2group.apollo.servlet.SSOAuthenticationFilter
    </filter-class>
</filter>

<filter-mapping>
    <filter-name>SSOAuthenticationFilter</filter-name>
    <url-pattern>/services/InfoService</url-pattern>
</filter-mapping>
```

In the login configuration section, add the following lines to define the client certificate authentication method:

```
<login-config>
    <auth-method>CLIENT-CERT</auth-method>
    <realm-name>WebRealm</realm-name>
</login-config>
```

Results

The i2 Analyze application is configured to allow client certificate authentication.

Redeploying i2 Analyze

Any time that you change the configuration of i2 Analyze, you must redeploy the system.

Procedure

1. Open a command prompt and navigate to `toolkit\scripts`.
2. Ensure that the application servers are stopped. To stop the application server, run the following command:

   ```
   setup -t stop
   ```

3. To deploy i2 Analyze components, run the following command:

   ```
   setup -t deploy
   ```

   Use the information in Troubleshooting the deployment process to ensure that i2 Analyze is deployed successfully.
4. To start the application server, open a command prompt and navigate to `toolkit\scripts`. Then, run the following command:

   ```
   setup -t start
   ```

5. Start, or restart, the HTTP server that hosts the reverse proxy.

Installing client certificates

After you create the client certificates, install the client certificates on the client workstations. After the client certificates are installed, they are available to be selected by users to use to log in to i2 Analyze.

About this task

On each client workstation that you want a user to be able to log in from, install the client certificate and exported keys as a Personal certificate. You can install multiple certificates and their associated keys on a workstation to allow multiple users to log in.

If you are not using a self-signed certificate, install the signer certificate that is associated with your client certificate so that your operating system can verify the client certificates.

Procedure

1. Copy your client certificate and the exported keys to a permanent directory on the client workstation.
2. Install the client certificate and exported keys to the **Personal** store:
   a) Double-click the keys file.
   For example, `Jenny.p12`.
   The **Certificate Import Wizard** is displayed.
   b) Click **Next**.
   c) Click **Browse**, and locate the keys file to import.
   For example, `Jenny.p12`. Then, click **Next**.
   d) Enter the password that you specified when the keys were exported from the key database, and click **Next**.
   e) Click **Place all certificates in the following store**.
   f) Click **Browse**, and select **Personal**.
   g) Click **Next**, and then click **Finish**.
3. If you are using a self-signed certificate, install the client certificate to the **Trusted Root Certification Authorities** store:
   a) Double-click the client certificate file.
   For example, `Jenny.der`.
   b) Click **Install Certificate**, and then click **Next**.
   c) Click **Place all certificates in the following store**.
   d) Click **Browse**, and select **Trusted Root Certification Authorities**.
   e) Click **Next**, and then click **Finish**.
   f) If the operating system cannot verify the certificate, a security warning is displayed.
   Click **Yes** to accept the certificate.

Results

The client certificate and exported keys are installed. Users can select a certificate to use to log in to i2 Analyze.

Testing the deployment

To ensure that the deployment is using client certificate authentication, access i2 Analyze from a client workstation. Connect to the system by using a web browser or Analyst's Notebook Premium.

Before you begin

- A client certificate must be installed on the client workstation. For more information about installing the client certificates, see “Installing client certificates” on page 106.
- The application server must be running.

Connecting to the Intelligence Portal

You can use X.509 certificates to authenticate users in the Intelligence Portal. The connection to the i2 Analyze server is secured, and the user is authenticated with the server by the client certificate that is provided by the client.

Before you begin

If you are using the Mozilla Firefox browser, you must add your certificates to the Mozilla Firefox, because Mozilla Firefox does not use the operating system's certificate store. Import your client certificates into **Your Certificates** in Mozilla Firefox.

Procedure

1. Open a web browser, and navigate to `https://`*`host_name`*`/apollo` (where *host_name* is the fully qualified domain name or IP address of the HTTP server).

   You are prompted to provide a certificate.
2. Select the client certificate to use to authenticate with the server from the list that is displayed.

Results

When client certificate authentication is configured correctly, you are logged in to the Intelligence Portal as the user associated with the selected certificate.

Connecting to the Analysis Repository from Analyst's Notebook Premium

You can use X.509 certificates to authenticate users in i2 Analyze from Analyst's Notebook Premium. The connection to the i2 Analyze server is secured, and the user is authenticated with the server by the client certificate that is provided by Analyst's Notebook Premium.

About this task

When you first start Analyst's Notebook Premium, or if you change the connection to a repository, the application displays a window that contain connection details for the Analysis Repository.

Procedure

1. Enter the name of the repository, and the URL of the server to connect to.

   Use the HTTPS protocol to connect to the server. For example, `https://`*`host_name`*`/apollo` (where *host_name* is the fully qualified domain name or IP address of the HTTP server).
2. Select **Use client certificate** and click **Browse**. Then, choose the client certificate to use to authenticate with the server.
3. Click **OK**.

Results

When client certificate authentication is configured correctly, you are logged in to i2 Analyze as the user associated with the selected certificate.

Connecting to the Information Store from Analyst's Notebook Premium

You can use X.509 certificates to authenticate users in i2 Analyze from Analyst's Notebook Premium. The connection to the i2 Analyze server is secured, and the user is authenticated with the server by the client certificate that is provided by Analyst's Notebook Premium.

About this task

Connect to the Information Store, and select a client certificate to use to authenticate with the i2 Analyze server.

Procedure

1. Enter the name of the repository, and the URL of the server to connect to.

   Use the HTTPS protocol to connect to the server. For example, `https://`*`host_name`*`/opal` (where *host_name* is the fully qualified domain name or IP address of the HTTP server).

   You are prompted to provide a certificate.
2. Select the client certificate to use to authenticate with the server from the list that is displayed.

Results

When client certificate authentication is configured correctly, you are logged in to i2 Analyze as the user associated with the selected certificate.

Resources for system protection

In order to protect your system from external forces, you must implement system controls that prevent or mitigate the effect of attacks. Although IBM does not manage login configuration, and the responsibility for protection of your network from external attack remains yours, the following communities provide a starting point for your investigation into preventative methods.

The Open Web Application Security Project

The Open Web Application Security Project Foundation is a not-for-profit organization that is dedicated to enabling organizations to conceive, develop, operate and maintain applications that can be trusted. In particular, see https://www.owasp.org/index.php/Blocking_Brute_Force_Attacks and https://www.owasp.org/index.php/Authentication_Cheat_Sheet#Password_Complexity.

SANS Institute

The System-Admin, Audit, Network and Security Institute is the largest source for information security training and security certification in the world. It also develops, maintains, and makes available at no cost, the largest collection of research documents about various aspects of information security, and it operates the Internet's early warning system - the Internet Storm Center. In particular, see https://www.sans.org/security-resources/policies/general/pdf/password-construction-guidelines

Common Weakness Enumeration

CWE™ is a community-developed list of common software security weaknesses. It serves as a common language, a measuring stick for software security tools, and as a baseline for weakness identification, mitigation, and prevention efforts.

In particular, see http://cwe.mitre.org/top25/index.html#CWE-307.

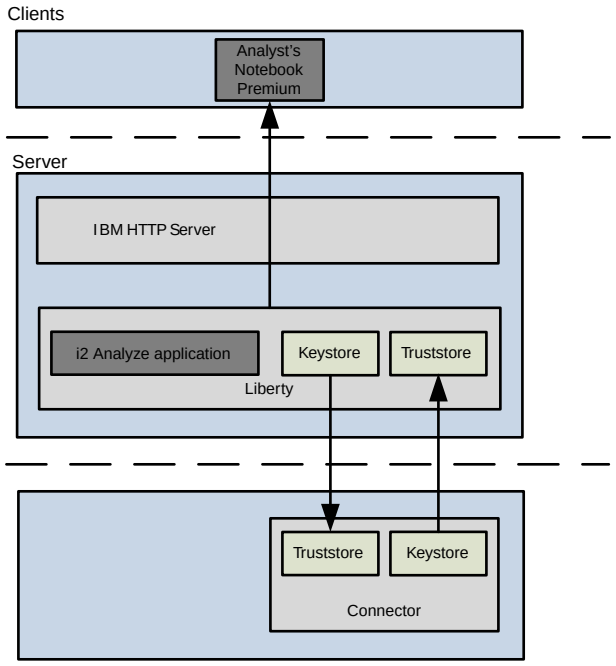
Client authenticated Secure Sockets Layer with IBM i2 Connect

To secure the connection between Liberty and any IBM i2 Connect connectors, you must configure Liberty and your connectors to use SSL. If you are using SSL, i2 Analyze enforces client authenticated communication with a connector.

Before you begin

In a production deployment you should configure i2 Analyze to connect to your connector using client authenticated SSL communication. To do so, your connector and i2 Analyze must trust the certificates that they receive during the SSL handshake process. In a production environment, the certificates must be signed by a trusted certificate authority. For more information about client authenticated SSL, see Client-authenticated TLS handshake.

The following diagram shows the keystores and truststores that are required for Liberty and the connector.



The Liberty server requires a keystore file and a truststore file, both (`*.jks`). Your connector can use any implementation for its keystore and truststore. The certificates in each truststore must trust the certificates received from the corresponding keystore.

The certificates that are required are as follows, where certificate authority (CA) X issues the certificates to the connector (the server) and Liberty (the client):

The connector requires:

- In its keystore:
  - The personal certificate issued to the connector by CA X
  - The connector's private key
- In the truststore:

- The CA certificate for CA X

Liberty requires:

- In its keystore:
  - The personal certificate issued to Liberty by CA X
  - Liberty's private key
- In its truststore:
  - The CA certificate for CA X

About this task

After you have created and populated the keystores and truststores for the connector and Liberty, you must configure Liberty to use SSL to communicate with any connectors.

The following steps explain the process of updating the i2 Analyze configuration with the location of a keystore and truststore to use, and the passwords that are used to access the certificates that are contained within them.

To configure the example-connector to use client authenticated SSL, and for examples of how to create keystores, truststore, and certificates for Liberty, follow the instructions in "Securing the example connector" on page 111.

Procedure

1. In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
   a) Add the `<key-stores>` element as a child of the `<application>` element. Then, add child `<key-store>` elements.

   For your keystore, specify the `type` as `key-store`, and `file` as the full path to your keystore. For your truststore, specify the `type` as `trust-store`, and `file` as the full path to your truststore.

   For example, add the attribute as highlighted in the following code:

```
<application http-server-host="true"
  name="opal-server" host-name="hostname">
...
    <key-stores>
        <key-store type="key-store"
          file="C:/IBM/i2analyze/i2-liberty-keystore.jks"/>
        <key-store type="trust-store"
          file="C:/IBM/i2analyze/i2-liberty-truststore.jks"/>
    </key-stores>
...
</application>
```

   b) Update the `base-url` attribute of any connectors using SSL to use the HTTPS protocol.

   For example:

```
<connectors>
    <connector id="example-connector" name="Example"
      base-url="https://localhost:3700/" />
</connector>
```

Note: Ensure that the host name that is used in the base URL matches the common name on the certificate of the connector.

2. Specify the keystore and truststore passwords in the credentials file.
   a) In a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
   b) Enter the password for the keystore and truststore that you specified in the `topology.xml` file.

```
ssl.keystore.password=password
ssl.truststore.password=password
```

3. Redeploy i2 Analyze so that the configuration changes are deployed to the application. For more information, see Deploying i2 Analyze.

What to do next

You can create your own connectors to use with the deployment of i2 Analyze, when you create your own connector you can implement security that conforms to the security required by IBM i2 Connect. For more information about creating your own connectors, see IBM i2 Analyze and i2 Connect.

When you use a connector configured for SSL communication, you should not see any warnings displayed in Analyst's Notebook Premium.

Securing the example connector

In a production deployment, you must configure your connectors with client authenticated SSL communication. You can configure the example connector to communicate with i2 Analyze using client authenticated SSL communication.

Before you begin

You must configure your connectors to use client authenticated SSL communication with i2 Analyze. For more information about configuring security between connectors and i2 Analyze, see “Client authenticated Secure Sockets Layer with IBM i2 Connect” on page 109.

About this task

Configure secure communication between Liberty and the example connector that is provided with i2 Analyze. The following steps demonstrate how to create the keystores, truststores, and certificates, then configure i2 Analyze and the example connector to use them.

When you create your own connector, you can implement the security implementation in the same way as used in the example connector, or a different method depending on your requirements.

In a production deployment, you must use a certificate that has been signed by a trusted certificate authority, which the connector can verify. The following steps use a self-signed certificate to demonstrate the functionality.

Procedure

Create a keystore for Liberty that contains a signed certificate that is used to authenticate with the connector.

1. Create a keystore and self-signed certificate for Liberty by using the Java keytool utility.

   For more information about the Java keytool utility, see keytool - Key and Certificate Management Tool.

   a) Open a command prompt and navigate to the `i2analyze\deploy\java\bin` directory.

b) Create the keystore and certificate by running the following command:

```
keytool -genkeypair -alias "libertyKey"
-keystore "C:\IBM\i2analyze\i2-liberty-keystore.jks"
-dname "CN=hostname" -keyalg RSA -storepass "libertyKeyStorePassword"
```

Important: Ensure that you enter values as follows:

- Set the location of the keystore to the directory that contains the toolkit. In some deployments, this path might be `C:\IBM\i2EIA`.
- Set the value of `CN` to the host name of the server that hosts Liberty as defined in the `topology.xml` file.

The password that is specified is used to access the keystore.

c) The example connector implementation requires the Liberty certificate in a base64 encoded format. Export the certificate in base64 encoding by running the following command:

```
keytool -exportcert -alias "libertyKey"
-keystore "C:\IBM\i2analyze\i2-liberty-keystore.jks"
-file "C:\IBM\i2analyze\i2-liberty-certificate.cer"
-rfc -storepass "libertyKeyStorePassword"
```

Create a keystore for the connector that contains a signed certificate that is used to authenticate with Liberty.

2. Create a keystore and certificate for the connector using the IBM Key Management Utility command-line interface.

   For more information about the IBM Key Management Utility, see Key Management Utility command-line interface (gskcmd) syntax.

   a) In a command prompt, navigate to the `IBM\HTTPServer\bin` directory.

   b) Create a keystore in PKCS12 format. To create the keystore, run the following command:

```
gskcmd -keydb -create -type pkcs12
-db "C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-
keystore.p12"
-pw "connectorKeyStorePassword"
```

   The password that is specified is used to access the keystore.

   c) Create a self-signed certificate by running the following command:

```
gskcmd -cert -create
-db "C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-
keystore.p12"
-label "connectorKey" -dn "CN=localhost" -pw "connectorKeyStorePassword"
```

   Important: Set the value of `CN` to the host name of the server that hosts the connector as defined in the `topology.xml` file. By default in the `daod-opal` example configuration, the value is `localhost`.

3. The example connector implementation requires the private key in the PEM format, and the certificate for the connector.

a) Export the personal certificate and private key into a PKCS12 file by running the following command:

```
gskcmd -cert -export
-db "C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-keystore.p12"
-pw "connectorKeyStorePassword" -label "connectorKey" -type pkcs12
-target "C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-key.p12"
-target_pw "connectorKeyPassword" -target_type pkcs12
```

b) Convert the private key from the PKCS12 format to the PEM format. You can use OpenSSL to do this, for more information about OpenSSL see https://www.openssl.org/source/. If you are using OpenSSL, you can run the following command:

```
openssl pkcs12 -in C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-key.p12
-out C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-key.pem
-nocerts -nodes
```

c) Extract the certificate for the connector by running the following command:

```
gskcmd -cert -extract
-db "C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-keystore.p12"
-label "connectorKey"
-target "C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-certificate.cer"
-pw "connectorKeyStorePassword"
```

Create a truststore for Liberty that contains the certificate that is used to trust the certificate that it receives from the connector.

4. Create the Liberty truststore and populate it with the connector's certificate using the Java keytool by running the following command:

```
keytool -importcert -alias "exampleconnector"
-keystore "C:\IBM\i2analyze\i2-liberty-truststore.jks"
-file "C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-certificate.cer"
-storepass "libertyTrustStorePassword"
```

   In response to the prompt, enter `yes` to trust the certificate.

5. Populate the connector keystore with Liberty's certificate using the IBM Key Management Utility by running the following command:

```
gskcmd -cert -add -type pkcs12
-db "C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-keystore.p12"
-pw "connectorKeyStorePassword" -label "libertyKey" -trust enable
-file "C:\IBM\i2analyze\i2-liberty-certificate.cer"
```

6. The example connector implementation requires the certificate that is used to trust Liberty to also be present in the `example-connector` directory. Copy the `C:\IBM\i2analyze\i2-liberty-certificate.cer` file to the `C:\IBM\i2analyze\toolkit\examples\connectors\example-connector` directory.
7. Start, or restart, the HTTP server that hosts the reverse proxy.

The certificates are now in place to enable client authentication SSL between Liberty and the connector.

8. Configure the example connector to reference the certificates that you created, and the host name of the gateway. Using a text editor, modify the `security-config.json` file with the following values.

   `https`

   Set to `true` to use the HTTPS protocol when connecting to the connector.

   `keyFileName`

   The file name for the private key of the connector in PEM format. For this example, `example-connector-key.pem`.

   `keyPassphrase`

   The password that is required to access the key file specified in `keyFileName`. For this example, `password`.

   `certificateFileName`

   The file name for the certificate of the connector. For this example, `example-connector-certificate.cer`.

   `certificateAuthorityFileName`

   The file name of the certificate that enables trust of the certificate received from Liberty. For this example, `i2-liberty-certificate.cer`.

   `gatewayCN`

   The common name of the gateway. This must be the value of the common name in the certificate the connector receives from Liberty. You specified the value of the CN in the certificate in step 1.

Configure Liberty to use the keystore and truststore that you created.

9. In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
   a) Add the `<key-stores>` element as a child of the `<application>` element. Then, add child `<key-store>` elements.

   For your keystore, specify the `type` as `key-store`, and `file` as the full path to your keystore. For your truststore, specify the `type` as `trust-store`, and `file` as the full path to your truststore.

For example, add the attribute as highlighted in the following code:

```
<application http-server-host="true"
  name="opal-server" host-name="hostname">
...
    <key-stores>
        <key-store type="key-store"
          file="C:/IBM/i2analyze/i2-liberty-keystore.jks"/>
        <key-store type="trust-store"
          file="C:/IBM/i2analyze/i2-liberty-truststore.jks"/>
    </key-stores>
...
</application>
```

b) Update the `base-url` attribute of any connectors using SSL to use the HTTPS protocol.

For example:

```
<connectors>
    <connector id="example-connector" name="Example"
      base-url="https://localhost:3700/" />
</connector>
```

Note: Ensure that the host name that is used in the base URL matches the common name on the certificate of the connector.

10. Specify the keystore and truststore passwords in the credentials file.
    a) In a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
    b) Enter the password for the keystore and truststore that you specified in the `topology.xml` file.

```
ssl.keystore.password=libertyKeyStorePassword
ssl.truststore.password=libertyTrustStorePassword
```

11. Redeploy i2 Analyze so that the configuration changes are deployed to the application. For more information, see Deploying i2 Analyze.
12. Start the example connector.
    a) In a command prompt, navigate to the `toolkit\examples\connectors\example-connector` directory.
    b) To start the Node.js server, run the following command:

```
npm start
```

Note: The example connector uses port number 3700. Ensure that no other processes are using this port number before you start the connector.

13. Start i2 Analyze.
    a) On the i2 Analyze server, open a command prompt and navigate to the `toolkit\scripts` directory.
    b) To start i2 Analyze, run the following command:

```
setup -t start
```

14. Start, or restart, the HTTP server that hosts the reverse proxy.

What to do next

Use Analyst's Notebook Premium to connect to your deployment. For more information, see Connecting IBM i2 Analyst's Notebook Premium to IBM i2 Analyze.

Now that the example connector is configured for SSL, no warnings are displayed in Analyst's Notebook Premium.

Configuring search options

Depending on the requirements of the environment that you are deploying into, a number of configurable options are available for searching and indexing. For example, for performance reasons, you can modify the minimum number of characters to be used within a wildcard search or customize alternative terms that are used.

Configuring the i2 Analyze Opal search

Data can be searched using a number of configurable options. With the Opal services, you can configure indexing, Quick Search, and Visual Query features to match your needs.

Configuring the Opal search index

i2 Analyze enables searching of information that is stored in the Information Store and other data sources. The quality of the search results is affected by the index that the Solr component creates, and the lists of synonyms that it maintains for common search terms.

By default, the Solr search index is created in US English, and uses synonym lists that contain US English terms. When you change the language of the search index, a default synonyms list in the specified language is used to create the associated term index. Solr provides the facility to configure the synonyms that are used for querying textual data. In i2 Analyze, you can use this option to apply a customized list of synonyms at query time.

A synonym is a word or phrase that means exactly or nearly the same thing as another word or phrase. Synonyms, if not accounted for, can cause a reduction in the relevance of a search result when you search for keywords that are present in alternative forms in your index.

The synonyms file is the part of the Solr configuration that accounts for the presence of synonyms in your data. For example, your data might contain the words, “bag, handbag, pocketbook, purse” for the concept “bag”. When someone searches they are likely to search for one, but expect results for all four. To meet that expectation, you might want to create a customized synonyms file to accommodate similar variations that are specific to your data. The exact words in a synonyms list that are most useful in your deployment depend on the content of your data. You can also use a mix of languages, which might be useful in some contexts, for example names: 'George, Γεώργιος, Jorge'.

The topology file is used to specify the language of the search index and to specify a customized synonyms file. Both the language country-code and the customized synonyms file specification are optional.

- If the language country-code and the synonyms file are not specified, then the default US English search index and associated default synonyms file are used.
- If only the language country-code is supplied, then the specified language country-code index and associated default synonyms file are used.
- If only the synonyms file is specified, then a US English index with the specified synonyms file is used.
- If both the language country-code and the synonyms file are specified, then the specified language and the specified synonyms file are used.

Specifying the Solr index language

By default, the Solr search index is created in US English. If you need to, you can change the language of the generated indexes.

About this task

Specification of the language country-code is optional. The default language value is `en_US` for US English. All indexes use the language country-code that is specified. The supported language options are `en_US` (US English), `ar_EG` (Arabic), and `he_IL`(Hebrew).

When you change the language, a default synonyms file is associated with the language selection. You can change the default synonyms file to use a customized synonyms file. For more information, see "Creating a Solr synonyms file" on page 118.

Procedure

1. In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
2. In the `<solr-cluster>` or `<solr-collection>` element, add the language-country-code attribute with the value for your chosen language.
   For example:

```
<solr-clusters>
    <solr-cluster language-country-code="ar_EG" ...>
      <solr-collections>
        <solr-collection="type_index" language-country-code="ar_EG" ... />
             </solr-collections>
      <solr-nodes/>
    </solr-cluster>
  </solr-clusters>
```

   Where *type* is the type of index to be used: `main` or `daod`.

   Note: You can set the language for the `main_index` or the `daod_index`, but you cannot have both in the same deployment.
3. Save the `topology.xml` file.
4. On the i2 Analyze server, open a command prompt and go to the directory `toolkit\scripts`.
5. To stop Websphere Application Server Liberty, run the following command.

```
setup -t stopLiberty
```

6. To create and upload the Solr configuration to the ZooKeeper hosts, run the following command.

```
setup -t createAndUploadSolrConfig
```

7. To clear the search index, run the following command.

```
setup -t clearSearchIndex
```

8. To start Websphere Application Server Liberty, run the following command.

```
setup -t startLiberty
```

Results

- When a language country-code is specified only in the `<solr-cluster>` element, all indexes use the language country-code specified.
- When a language country-code is specified in the `<solr-collection>` element, only the index in that collection uses the language that is specified, this value overrides the language country-code specified in `<solr-cluster>`.

Creating a Solr synonyms file

The deployment toolkit specifies the synonyms file that contains the list of alternative terms for each supported language. You can modify or replace the supplied default files to introduce extra terms that match the needs of your deployment.

About this task

The default synonyms file and synonyms list are in US English. The synonyms files that are associated by default with each supported language are supplied in the directory, `toolkit/examples/solr-resources/solr-synonyms`.

To customize the alternative terms that are used in search operations for your data, you can create files that contain different terms from those terms that are contained in the supplied synonyms files.

The topology file can be used to specify a customized synonyms file as an alternative to the default synonyms file that is associated with the language country code. The customized file must adhere to the following guidelines.

- The file must be UTF-8 encoded.
- The terms in the file must match the terms that are produced by the analyzer chain that is used in Solr prior to the synonym filter being applied.
- If multiple forms of a word exist, all the forms must be specified in order for synonym matching to work on each form.
- Words from Latin script languages, for example French or Italian, must be specified without diacritics. For example, use the following substitution:
  - a instead of á
  - c instead of ç
- Arabic and Hebrew words must be specified exactly as they are written.

Procedure

1. Create a text file that defines synonyms in the required Solr format.

   For more information, see https://lucene.apache.org/core/6_6_2/analyzers-common/org/apache/lucene/analysis/synonym/SolrSynonymParser.html.

   Note:

   a. You cannot search for multi-word terms. However, if you have data that contains terms "USA" and "United States of America", you can search for "USA" and use a synonym to ensure a match with "United States of America".

   b. You can provide synonyms for terms that include punctuation. However, a search on such a term might not work correctly. The unexpected result is because a filter is applied before synonyms, which means, for example, "Mary-Ann" becomes "Mary,Ann" and then synonyms are expanded from "Mary and "Ann"; not "Mary-Ann" or "Maryann".
2. Save the file with a `.txt` extension, for example `custom-synonym.txt`.

3. In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
4. In the `<solr-cluster>` or `<solr-collection>` element, add the location of your synonyms file. For example:

```
<solr-clusters>
    <solr-cluster synonyms-file="C:/custom-synonyms.txt" ...>
      <solr-collections>
              <solr-collection id="type_index" synonyms-file="C:/custom-synonyms.txt" ... />
      </solr-collections>
      <solr-nodes/>
    </solr-cluster>
  </solr-clusters>
```

   Where *type* is the type of index to be used: `main` or `daod`.

   Note: You can set the synonyms file for the `main_index` or the `daod_index`, but you cannot have both in the same deployment.
5. Save the `topology.xml` file.
6. On the i2 Analyze server, open a command prompt and go to the directory `toolkit\scripts`.
7. To create and upload the Solr configuration to the ZooKeeper hosts, run the following command.

```
setup -t createAndUploadSolrConfig
```

Results

- When a synonym file is specified only in the `<solr-cluster>` element, all indexes use the synonym file that is specified.
- When a synonym file is specified in the `<solr-collection>` element, only the index in that collection uses the synonym file that is specified, this value overrides the synonym file that is specified in the `<solr-cluster>`.

Setting up search results filtering

In i2 Analyze, users can filter search results. You can configure the types of items and properties, as well as metadata criteria, that appear in the filter list by creating and configuring facets.

About this task

In a deployment of i2 Analyze, the results configuration file defines which property types and metadata criteria for each item type to display with facets in both the Quick Search and Visual Query results views. Both the Quick Search and Visual Query results views display the same filters.

When you deploy the `information-store-opal` example, the `law-enforcement-schema-results-configuration.xml` results configuration file that configures the facets specific to the `law-enforcement-schema.xml` is used. In addition, an example configuration file for each of the example schemas is included in the `examples\schemas` directory. If your system uses a modified version of one of the example schemas, you can modify the appropriate results configuration file. The example results configuration files are located the `configuration\examples\schemas\en_US` directory. The results configuration file that you want to use in the deployment is located in the `toolkit \configuration\fragments\common\WEB-INF\classes` directory.

In your results configuration file, the item types that are defined must correspond to entity and link types in your schema otherwise you are unable to start i2 Analyze. Examine your schema and the data in your system, and decide which property types for each item type to display with facets in the results view. Metadata criteria are not defined in the schema, however for each item type you can decide which metadata criteria to display with facets.

If you do not specify a results configuration file, all of the property types and metadata criteria that can be displayed with facets, for all of the item types, are displayed in the results view in schema order.

Selecting the property types that can be used to filter search results can improve the faceting performance, and improve the facets that are displayed to a user. For example, if a property type contains only unique values, the user sees a facet for each value that is returned in search results. If you configure that property type to not display with facets, it means that fewer facets must be calculated and that more useful facets from different property types are displayed instead. Examples of property types that you might configure to not display with facets are license plates and social security numbers.

For more information about the results configuration file and the changes that you can make, see "Understanding the results configuration file" on page 120.

Procedure

1. Using an XML editor, open the results configuration file that you would like to modify.
2. Add, modify, or remove any `<ItemTypeFacet>` elements and appropriate child `<PropertyTypeFacet>` and `<MetadataFacet>` elements for your deployment.

   Note: Property types that have the following logical types cannot be used to filter search results:

   - `GEOSPATIAL`
   - `MULTIPLE_LINE_STRING`
3. Save and close the file.

   Note: Ensure that your modified file is stored in the `toolkit\configuration\fragments\common\WEB-INF\classes` directory.
4. If you have changed the name of the file that is used in the original deployment, you must reset the name of the file that the deployment uses.
   a) Using a text editor, open the `DiscoServerSettingsCommon.properties` file in the `toolkit\configuration\fragments\opal-services-is\WEB-INF\classes` directory.
   b) Ensure that the value of the `ResultsConfigurationResource` property is set to the name of your results configuration file, then save and close the file.

      Tip: If you do not want to configure your results, clear the value of the `ResultsConfigurationResource` property.

What to do next

Redeploy i2 Analyze, and run a selection of queries on an Information Store. Continue to change the configuration until you are satisfied with the types of filter that are available.

Understanding the results configuration file

In your results configuration file, you define the item types and associated property types and metadata criteria that can be used to filter results for each of the entity and link types in your i2 Analyze schema. By defining the filtering behavior for different item types, you can refine the filtering to provide the most valuable filters for your environment.

The results configuration file has the following basic shape:

```
<!-- Results configuration for Law_Enforcement_Schema.xml -->
<tns:ResultsConfiguration ...>
    <Facets InlineLimit="5" ViewAllLimit="100">
        <!-- Address -->
        <ItemTypeFacet TypeId="ET1" Subfacets="ExcludeSpecific">
            <!-- Unique Reference -->
            <PropertyTypeFacet TypeId="ADD1" />
            ...
            <MetadataFacet Criterion="NotesCreatedBy" />
            ...
        </ItemTypeFacet>
        ...
    </Facets>
</tns:ResultsConfiguration>
```

Where the <ItemTypeFacet> element is used to define which item type you are defining property types and metadata criteria for. The value of the TypeID attribute specifies the item type. This value corresponds to the Id value of an <EntityType> or <LinkType> in the schema.

Property types are specified in child <PropertyTypeFacet> elements. The value of the TypeId attribute specifies the property type. This value corresponds to the Id value of a <PropertyType> in the i2 Analyze schema.

Metadata criteria are specified in child <MetadataFacet> elements. The value of the Criterion attribute specifies the metadata criteria and can have the following values:

NotesCreatedBy

The display name of the user who has added a note to the record.

FirstUploadedBy

The display name of the user who first uploaded the record.

FirstUploaded

The date that the record was first uploaded.

LastUploadedBy

The display name of the user who last uploaded the record.

LastUploaded

The date that the record was last uploaded.

IngestionDataSourceName

The data source name that the record was ingested from.

If the record was uploaded from Analyst's Notebook Premium, the IngestionDataSourceName is automatically set to ANALYST.

Note: Any child <MetadataFacet> elements must be specified after all of the child <PropertyTypeFacet> elements within an <ItemTypeFacet> element.

In the <ItemTypeFacet> element, the value of the Subfacets attribute defines the method for specifying the property types and metadata criteria.

The Subfacets attribute can have the following values:

All

All property types of this item type and metadata criteria are available as filterable options for the results. This behavior is the default if an item type is not added to the results configuration file. For example:

```
<ItemTypeFacet TypeId="ET5" Subfacets="All" />
```

Declaring this fragment while working with the law enforcement schema will allow you to filter by 'Person' and by all the available properties and metadata.

Note: You must not specify any child elements when the value of the Subfacets attribute is All.

IncludeSpecific

Specific property types of this item type and metadata criteria are displayed in the filtering lists. For example:

```
<ItemTypeFacet TypeId="ET5" Subfacets="IncludeSpecific">
    <PropertyTypeFacet TypeId="PER4" />
    <PropertyTypeFacet TypeId="PER6" />
    <MetadataFacet Criterion="NotesCreatedBy" />
</ItemTypeFacet>
```

Declaring this fragment while working with the law enforcement schema will allow you to filter by 'Person' and by 'First (Given) Name', 'Family Name', and 'NotesCreatedBy'.

Note: You must specify at least one child <PropertyTypeFacet> or <MetadataFacet> element when the value of the Subfacets attribute is IncludeSpecific.

ExcludeSpecific

Specific property types of this item type and metadata criteria are excluded from the filtering lists. For example:

```
<ItemTypeFacet TypeId="ET3" Subfacets="ExcludeSpecific">
    <PropertyTypeFacet TypeId="VEH2" />
    <MetadataFacet Criterion="FirstUploaded" />
</ItemTypeFacet>
```

Declaring this fragment while working with the law enforcement schema will allow you to filter by 'Vehicle', but not by 'License Plate Number' or 'FirstUploaded'.

Note: You must specify at least one child <PropertyTypeFacet> or <MetadataFacet> element when the value of the Subfacets attribute is ExcludeSpecific.

None

No property types of this item type and metadata criteria are available for filtering. For example:

```
<ItemTypeFacet TypeId="ET5" Subfacets="None" />
```

Declaring this fragment while working with the law enforcement schema will allow you to filter by 'Person' but not by any of the properties of the Person type, such as eye color nor any of the metadata criteria.

Note: You must not specify any child elements when the value of the Subfacets attribute is None.

Additionally, you can disable all property type and metadata criteria for faceting by using the `PropertyTypeFacetsEnabled` and `MetadataFacetsEnabled` attributes of the `<Facets>` element. By default, both property type and metadata criteria are enabled for faceting. To disable, set the attribute values to `false`.

Modifying the wildcard minimum character limits

Wildcard characters are used to match zero or more alphanumeric characters in a search term. You can configure how users complete Quick Search and Visual Queries by modifying the minimum number of characters that must be included in a search term that includes wildcards.

About this task

The following wildcard characters are available in i2 Analyze:

*****

Matches zero or more alphanumeric characters in this position.

For example, the search term `Tim*` matches `Tim`, `Time`, and `Timely`.

?

Matches one alphanumeric character in this position.

For example, the search term `Tim?` matches `Time`.

Wildcard characters can be used at any position in a search term.

Search terms with wildcard queries might result in a large number of matches, which might cause performance problems. To reduce the possible matches from a wildcard search, you can ensure that users provide a minimum number of characters with a wildcard. For example, the term `"*"` matches everything. If users must provide a minimum of 3 characters with an asterisk, for example the term `"abc*"`, the number of matches is reduced to values that begin with "abc".

If the minimum number of characters is set too high, users might not be able to search for the terms that they need. For example, in a deployment where the wildcard minimum characters are configured as follows:

- A minimum of 3 characters other than asterisks (*) must be provided in a term.
- A minimum of 2 characters other than question marks (?) and asterisks (*) must be provided in a term.

If the user knows only 2 characters of a license plate, they might not be able to use a wildcard search:

- If the user knows the position of the 2 characters, they can use the question mark wildcard character in the search. If the 2 characters are in positions 2 and 3, then the query `"?AB*"` is valid in the configuration that is described.
- If the position of the characters is not known, the user might want to search for `"*AB*"`, which is invalid in the configuration that is described.

In addition to wildcard characters that are specifically entered as part of Visual Query, several conditions provide implicit wildcard logic:

- 'Starts with' - Applies an asterisk to the end of the condition. For example, 'Starts with: Fred' is equivalent to 'Fred*', which could match; Fred, Frederick, or Freddie.
- 'Ends with' - Applies an asterisk to the start of the condition. For example, 'Ends with: Fred' is equivalent to '*Fred', which could match; Fred, Wilfred, or Alfred.
- 'Contains' - Applies an asterisk to both the start and the end of the condition. For example, 'Contains: Fred' is equivalent to '*Fred*', which could match any of the above terms, but also include Alfredo.

The use of these conditions follow the same limits as wildcard characters that have been entered explicitly.

To change the minimum number of characters that must be included in a search query with a wildcard character, edit properties in `DiscoServerSettingsCommon.properties`.

The properties that specify the minimum number of characters for Quick Search are:

`WildcardMinCharsWithAsterisk`

The minimum number of characters other than asterisks (*) that must be included in a wildcard query that contains an asterisk.

`WildcardMinCharsWithQuestionMark`

The minimum number of characters other than question marks (?) and asterisks (*) that must be included in a wildcard query that contains a question mark. This value should be less than, or equal to the value of the `WildcardMinCharsWithAsterisk` property.

The properties that specify the minimum number of characters for Visual Query are:

`VisualQueryWildcardMinCharsWithAsterisk`

The minimum number of characters other than asterisks (*) that must be included in a Visual Query condition that contains or implies asterisks.

`VisualQueryWildcardMinCharsWithQuestionMark`

The minimum number of characters other than question marks (?) and asterisks (*) that must be included in a wildcard query that contains a question mark. This value should be less than, or equal to the value of the `VisualQueryWildcardMinCharsWithAsterisk` property.

Procedure

1. Using a text editor, open the `DiscoServerSettingsCommon.properties` file. You can find this file in the following location: `toolkit\configuration\fragments\opal-services-is\WEB-INF\classes`.
2. For Quick Search, edit the values of the `WildcardMinCharsWithAsterisk` and `WildcardMinCharsWithQuestionMark` properties.
3. For Visual Query, edit the values of the `VisualQueryWildcardMinCharsWithAsterisk` and `VisualQueryWildcardMinCharsWithQuestionMark` properties.
4. Save and close the file.

What to do next

After you complete the instruction in “Redeploying and resetting i2 Analyze” on page 148 to redeploy i2 Analyze, run a selection of Quick Search and Visual Queries that use conditions including wildcard characters.

Visual Query condition restrictions

In order to create a configuration file that helps you to manage your Visual Query performance effectively, it is important to understand the available restrictions. Having a proper understanding of the implications of the rules that can be created, allows you to reduce the number of iterations required.

The `visual-query-configuration.xml` file in `configuration/fragments/opal-services-is/WEB-INF/Classes` includes commented out examples of rules applicable to the law enforcement schema. Using these examples as guidance, you can create rules that apply to your system.

Important: The rules that are added in `visual-query-configuration.xml` are applied to the i2 Analyze server and not the Analyst's Notebook Premium client. As such, users are still able to construct queries that include restricted conditions, but the server will prevent the query from being run.

Depending on your circumstances, there are two main approaches that can be used to create your rules:

Invalidating a particular type of query

If you are aware of query types that are an issue, you can deny the specific conditions that lead to the issue, allowing all other Visual Query behavior to remain unchanged.

Enabling specific queries

If you are uncertain about the type of queries that might produce an issue, or want to restrict the types of searches that can be carried out, you can deny all conditions, and selectively allow the types of query that you require.

As you build up your Visual Query configuration file, it is worth bearing in mind the following key concepts:

Rule components

Each rule consists of the following components:

- Rule Type - Whether the rule should allow or deny a specific type of operation
- Item Type - Which Item Type the rule should apply to
- Property Type - Which Property Types for the specified Item Type that the rule should cover
- Operator - Which Operator Types to apply the rule to
- Date and Time Aspects - Which types of temporal aspect to apply the rule to. For example, day of the week, or time of the day.

Implicit 'all valid'

If one or more of the above rule components are not specified, the rule will be applied to all the Visual Query conditions that would be valid.

Rule ordering

Visual Query rules are applied sequentially, meaning that if conflicting rules are added, they are handled in order, allowing rules that are later in the file to overwrite earlier rules. This allows you to refine your conditions, for example:

```
<Deny/>
<!--Allow 'Person' searches to include conditions for
'Date of birth' and 'Gender'
that are exactly or between a specified range  -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9, PER15"
Operators="EQUAL_TO, BETWEEN"/>
```

Using these rules, searches for people with specified dates of birth or genders are permitted, but all other Visual Query conditions are prevented from running.

Rule Type

There are two types of Visual Query condition rule. The type of restriction determines whether the operators specified in the rule are allowed or denied.

The type of rule can be either:

- Deny - Specify a rule that prevents an operation

- Allow - Specify a rule that enables an operation

Note: For each rule that you add to the `visual-query-configuration.xml`, you must specify the type of rule.

Item Type Identifier
A restriction that only includes the restriction type will apply to all the types of item that are specified in the schema. If required, to apply the restriction to a specific type of item, you can add an Item Type Identifier.

The `ItemTypeId` attribute allows you to specify an item type to restrict. It assumes that the specified value is found in the current i2 schema that is being used in your deployment. To help ensure that you values are valid, you might want to have the schema open for reference when you are creating your restrictions. In addition, to help troubleshoot issues, it may also help to add the display name or description of the item type into a comment about the restriction.

For example the following item type in the schema:

```
<EntityType Id="ET5"
SemanticTypeId="guid8A586959-9837-47DE-8DBF-BC7031F01545"
Description="Person details"
DisplayName="Person"
Icon="Person (Shaded Shirt)">
```

could be used to create the following rule in your configuration file:

```
<!--Allow 'Person' searches to include conditions
that are exactly or between a specified range  -->
<Allow ItemTypeId="ET5" Operators="EQUAL_TO, BETWEEN"/>
```

Note: You can only specify one type of item per rule.

Property Type Identifiers
If you have added an Item Type Identifier to a Visual Query rule, by default, the rule will apply to all the applicable property types. If required, to apply the rule to specific property types, you can add the property types identifiers.

The `PropertyTypeIds` argument allows you to specify the property types to restrict. It assumes that the specified values are found in the current i2 schema that is being used in your deployment. To help ensure that your values are valid, you might want to have the schema open for reference when you are creating your restrictions. In addition, to help troubleshoot issues, it may also help to add the display name or description of the property type into a comment about the rule.

Note: Property Type Identifiers can only be applied to rules that specify an Item Type, and the Property Types must correspond to the specified Item Type

For example the following property types in the law enforcement schema:

```
<EntityType Id="ET5"
SemanticTypeId="guid8A586959-9837-47DE-8DBF-BC7031F01545"
Description="Person details"
DisplayName="Person"
Icon="Person (Shaded Shirt)">
...
  <PropertyType Position="2"
  Mandatory="false"
  SemanticTypeId="guidFE45F1C4-B198-4111-8123-F42D2CD6419D"
  DisplayName="Date of Birth"
  Description=""
  LogicalType="DATE"
  Id="PER9">
    <PossibleValues />
  </PropertyType>
  <PropertyType Position="3"
   Mandatory="false"
   SemanticTypeId="guid7548369B-BA9A-4C4B-AEAD-0CB442EAFA27"
   DisplayName="Gender"
   Description=""
   LogicalType="SUGGESTED_FROM"
   Id="PER15">
     <PossibleValues>
       <PossibleValue Description="" Value="&lt;Unknown&gt;"/>
       <PossibleValue Description="Male" Value="Male"/>
       <PossibleValue Description="Female" Value="Female"/>
     </PossibleValues>
...
</EntityType>
```

could be used to create the following rule:

```
<!--Allow 'Person' searches to include conditions for
'Date of birth' and 'Gender'
that match on an exact value or a range of values  -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9, PER15"
Operators="EQUAL_TO, BETWEEN"/>
```

Note: This example restriction allows conditions to be run that search for 'equal to' exact values. In addition it includes allowing conditions that search between specified values. As a range of values cannot be determined for a 'suggested from' property type, conditions that are between a specified range will only be enabled for dates of birth.

Date and Time Aspects

When a rule specifies property types with the DATE or DATE_TIME data type, there are a number of options for querying that data. Rules can be applied to specific aspects of temporal information.

DATE_AND_TIME

Apply the rule to conditions that include both a date and time. For example: Searches for people spotted in a specific location at specific time.

DATE

Apply the rule to conditions that focus on a date. For example: Searches for people born on a particular day.

TIME

Apply the rule to conditions that focus on a time. For example: Searches for financial transactions that regularly occur at a set time.

DAY_OF_MONTH

Apply the rule to conditions that focus on the day of the month. For example: Searches for people paid on a specific day of the month.

MONTH

Apply the rule to conditions that focus on the month. For example: Searches for people born within a specific month.

QUARTER

Apply the rule to conditions that focus on the quarter of the year. For example: Searches for financial results.

YEAR

Apply the rule to conditions that focus on a specific year. For example: Searches for people born within a specific year.

DAY_OF_WEEK

Apply the rule to conditions that focus on a specific day of the week. For example: Searches for events that always occur on a Tuesday.

WEEK_OF_YEAR

Apply the rule to conditions that focus on the week of the year as calculated using the ISO week date system. For example weekly sales results.

Operators

You can use operators to specify the types of visual query condition to restrict. If you do not specify an operator, the restriction applies to all valid operations.

STARTS_WITH

Applies an implicit asterisk to the end of the condition. For example, `Starts with: Fred` is equivalent to 'Fred*', which might match; Fred, Frederick, or Freddie.

Supported logical types:

- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

Note: The server places a limit on the length of the strings that this operator considers. By default, the limit is 256 characters.

For example:

```
<!--Allow 'Person' searches to include conditions for
'First (Given) Name' that start in a given value  -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER4"
Operators="STARTS_WITH"/>
```

ENDS_WITH

Applies an implicit asterisk to the start of the condition. For example, `Ends with: Fred` is equivalent to '*Fred', which might match; Fred, Wilfred, or Alfred.

Supported logical types:

- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

Note: The server places a limit on the length of the strings that this operator considers. By default, the limit is 256 characters.

For example:

```
<!--Allow 'Person' searches to include conditions for
'First (Given) Name' that end in a given value  -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER4"
Operators="ENDS_WITH"/>
```

CONTAINS

Applies an implicit asterisk to both the start and the end of the condition. For example, 'Contains: Fred' is equivalent to '*fred*', which could match any of the above terms, but also include Alfredo.

Supported logical types:

- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

For example:

```
<!--Allow 'Person' searches to include conditions for
'First (Given) Name' that contain a given value  -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER4"
Operators="CONTAINS"/>
```

WILDCARD_PATTERN

An exact match to the specified term that includes wildcard characters. For example, `Wildcard pattern: Fr?d` is equivalent to 'Fr?d', which matches: Fred, but not Alfredo.

Supported logical types:

- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

For example:

```
<!--Allow 'Person' searches to include conditions for
'First (Given) Name' that match on a wildcard pattern  -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER4"
Operators="WILDCARD_PATTERN"/>
```

NOT_WILDCARD_PATTERN

Excludes an exact match to the specified term that includes wildcard characters. For example, `Not wildcard pattern: Fr?d` matches anything aside from: Fr?d.

Supported logical types:

- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

For example:

```
<!--Allow 'Person' searches to include conditions for
'First (Given) Name' that do not match on a wildcard pattern  -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER4"
Operators="NOT_WILDCARD_PATTERN"/>
```

EQUAL_TO

An exact match to the specified term.

Supported logical types:

- BOOLEAN
- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME
- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

Note: The server places a limit on the length of the strings that the operator considers. By default, the limit is 256 characters.

For example:

```
<!--Allow 'Person' searches to include conditions for
'Date of birth' and 'Gender'
that match on an exact value  -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9, PER15"
Operators="EQUAL_TO"/>
```

NOT_EQUAL_TO

Exact matches to the specified term should be excluded. For example, `Not equal to: Fred` matches anything aside from: Fred.

Supported logical types:

- BOOLEAN
- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME
- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

Note: The server places a limit on the length of the strings that the operator considers. By default, the limit is 256 characters.

For example:

```
<!--Allow 'Person' searches to include conditions for
'Date of birth' that do not match on an exact value  -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9"
Operators="NOT_EQUAL_TO"/>
```

GREATER_THAN

Matches values that are higher than a set value.

Supported logical types:

- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME

For example:

```
<!--Allow 'Person' searches to include conditions for
'Date of birth' that are greater than a specified value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9"
Operators="GREATER_THAN"/>
```

GREATER_THAN_OR_EQUAL_TO

Matches values that are higher than or equal to a set value.

Supported logical types:

- DECIMAL

- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME

For example:

```
<!--Allow 'Person' searches to include conditions for
'Date of birth' that are greater than or equal to a specified value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9"
Operators="GREATER_THAN_OR_EQUAL_TO"/>
```

LESS_THAN

Matches values that are less than a set value.

Supported logical types:

- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME

For example:

```
<!--Allow 'Person' searches to include conditions for
'Date of birth' that are lower than to a specified value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9, PER15"
Operators="LESS_THAN"/>
```

LESS_THAN_OR_EQUAL_TO

Matches values that are less than or equal to a set value.

Supported logical types:

- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME

For example:

```
<!--Allow 'Person' searches to include conditions for
'Date of birth' that are lower than or equal to a specified value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9"
Operators="LESS_THAN_OR_EQUAL_TO"/>
```

BETWEEN

Matches values that are within a set range.

Supported logical types:

- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME

For example:

```
<!--Allow 'Person' searches to include conditions for
'Date of birth' that match on a range of values  -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9"
Operators="BETWEEN"/>
```

IS_SET

Matches properties with any populated value.

Supported logical types:

- BOOLEAN
- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME
- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

For example:

```
<!--Allow 'Person' searches to include searches for 'Date of birth' values
that have been entered  -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9"
Operators="IS_SET"/>
```

IS_NOT_SET

Matches properties without a value entered.

Supported logical types:

- BOOLEAN
- DECIMAL
- DATE
- DATE_AND_TIME

- DOUBLE
- INTEGER
- TIME
- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

For example:

```
<!--Allow 'Person' searches to include searches for 'Date of birth' values
that have not been entered  -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9"
Operators="IS_NOT_SET"/>
```

Restricting Visual Query conditions

By default, all the Visual Query operators are available for all the record types that support them in your system. To prevent the creation of queries that take too long to produce results, you can configure Visual Query to restrict the operators that are valid in conditions.

Procedure

1. Set the rules that you would like to make to your Visual Query conditions:
   a) Using an XML editor, open the `visual-query-configuration.xml` file.
      You can find this file in the following location: `toolkit\configuration\fragments\opal-services-is\WEB-INF\classes`.
   b) Using the supplied examples, add the rules that you would like to implement.
      Note: Ensure that you add the restrictions paying attention to the ordering as later rules will override earlier rules if applicable.
   c) Save your changes.
2. Update the `DiscoServerSettingsCommon.properties` file to use your configuration rules.
   a) Using a text editor, open the `DiscoServerSettingsCommon.properties` file.
      You can find this file in the following location: `toolkit\configuration\fragments\opal-services-is\WEB-INF\classes`.
   b) Set the `VisualQueryConfigurationResource`
      For example: `VisualQueryConfigurationResource=visual-query-configuration.xml`
   c) Save your changes.

What to do next

After you redeploy i2 Analyze, run a selection of Visual Queries that use conditions.

Restricting the length of Visual Query lists

When a Visual Query is constructed by a user, certain conditions allow the values to be specified as a list of values. The length of these lists can be restricted to reduce the size of messages from the client.

About this task

By default, the number of items that can be used in a list condition for a visual query condition is set to 10000. If you reduce this value, the number of values that can be applied to conditions of this type is reduced.

Procedure

To modify the list length:

1. Using a text editor, open the `DiscoServerSettingsCommon.properties` file. You can find this file in the following location: `toolkit\configuration\fragments\opal-services-is\WEB-INF\classes`.
2. Edit the value of `VisualQueryMaxValuesInList`.
3. Save and close the file.

What to do next

After you complete the instruction in "Redeploying and resetting i2 Analyze" on page 148 to redeploy i2 Analyze, run a selection of Visual Queries that use conditions including lists.

Configuring the i2 Analyze Analysis Repository search

i2 Analyze enables searching of information that is stored in the Analysis Repository. The quality of the search results is affected by the index that the platform creates, and the lists of synonyms that it maintains for common search terms.

By default, i2 Analyze assumes that item data is in US English when it creates the search index, and uses synonym lists that contains US English terms. When you change the language of the search index, a default synonym list in the specified language is used to create the alternative term index.

Creating a custom alternative term index

The deployment toolkit contains files to create a list of alternative terms for every supported language. You can modify or replace the supplied files to introduce terms that match the needs of your deployment.

About this task

In i2 Analyze, three key components enable the alternative term functionality:

- A dictionary of terms that have alternatives
- A list of synonyms for each of the terms in the dictionary
- A list of terms that are actively ignored when the user attempts to search for them

The supplied dictionary and synonym list are in the `search-alternatives` directory of the deployment toolkit, at `configuration\environment\common\search-alternatives`.

To customize the alternative terms, you can create files that specify different terms from the supplied files.

Procedure

1. Within `toolkit\configuration\environment\common`, create the following directory structure:

```
- search-alternatives-input
      - my-alternatives
            - alternatives
```

   Where *my-alternatives* is a custom name.

   For example: `toolkit\configuration\environment\common\search-alternatives-input\my-alts\alternatives`.

   Tip: If you are configuring external data sources, you can create more than one directory in this location. For example, you might want to tailor the alternative term index differently for data from different sources.

2. Copy the contents of the existing `search-alternatives` directory into your new location.
3. Set up the dictionary files:
   a) Navigate to the `dictionary` directory inside your new location, and open the directory for the language that you want to customize.
   b) Create a text file that specifies the terms that will have alternatives, one per line.
   For example:

```
Bike
Car
Truck
Van
```

   c) Save the file with a `.txt` extension.

   Note: Ensure that the saved file is encoded in `UTF-8`.

4. Set up the synonym lists:
   a) Navigate to the `synonyms` directory inside your new location, and open the directory for the language that you want to customize.
   b) Create an XML file that defines synonyms in the following format:

```
<Synonyms VersionMajor="" VersionMinor=""
          VersionRelease="" VersionBuild="">
  <Entry Word="Term" Syn="Synonym1 Synonym2" />
</Synonyms>
```

   For example:

```
<Synonyms VersionMajor="1" VersionMinor="1"
          VersionRelease="57" VersionBuild="0">
  <Entry Word="BIKE" Syn="BICYCLE MOTORBIKE" />
</Synonyms>
```

   c) Save the file with a `.xml` extension.
5. Navigate to the `toolkit\configuration\environment\topology.xml` file, and open it in an XML editor.
6. For each `<lucene-index>` element in the file, specify the following attributes:

Attribute	Description
alternative-id	The location of the source files for your custom alternative terms. For example, alternative-id="my-alts" looks for the files in the following location: toolkit\configuration\environment\common\search-alternatives-input\my-alts\alternatives\. **Note:** This setting defaults to the standard files (if available) for the specified language-country-code.
stop-words	A comma-separated list of words to be filtered out of search conditions. For example: stop-words="a,an,and". **Note:** This setting defaults to the Lucene stop word list for the specified language-country-code.

7. Within each <war> fragment that uses an index, update the <lucene-index-id> to match the specified <lucene-index>.
   For example:

```
<lucene-index-ids>
  <lucene-index-id value="my-index" />
</lucene-index-ids>
```

8. Save your changes.

Results
When you run the deployment scripts, the alternative terms index is created with the specified settings.

Specifying the index language

By default, the deployment toolkit generates indexes and alternative terms in US English. When you need to, you can change the language of both the generated indexes and the alternative terms.

Procedure

1. Open topology.xml and locate the <lucene-index> element for the index that you want to set the language for.
2. Add an attribute of the form language-country-code="*xx_XX*", where *xx_XX* is the code for the language that you want to use.
   For example:

```
<lucene-index id="ar" main-index-location="C:/IBM/i2analyze/data/ar/main_index"
              alternatives-location="C:/IBM/i2analyze/data/ar/alternatives"
              language-country-code="fr_FR" />
```

   Note: Ensure that all file paths use forward slashes (/).
3. Save the topology.xml file.

   Note: Ensure that the saved topology.xml file is encoded in UTF-8.

Modifying the wildcard minimum character limit

Wildcard characters are used to match zero or more alphanumeric characters in a search term. You can configure the minimum number of characters that must be included in a search term that includes wildcards.

About this task

The following wildcard characters are available in i2 Analyze:

*****

Matches zero or more alphanumeric characters in this position.

For example, the search term `Tim*` matches `Tim`, `Time`, and `Timely`.

?

Matches one alphanumeric character in this position.

For example, the search term `Tim?` matches `Time`.

Wildcard characters can be used at any position in a search term.

Search terms with wildcard queries might result in a large number of matches, which might cause performance problems. To reduce the possible matches from a wildcard search, you can ensure that users provide a minimum number of characters with a wildcard. For example, the term `"*"` will match everything. If users must provide a minimum of 3 characters with an asterisk, for example the term `"abc*"`, the number of matches is reduced to values that begin with "abc".

If the minimum number of characters is set too high, users might not be able to search for the terms that they need. For example, in a deployment where the wildcard minimum characters are configured as follows:

- A minimum of 3 characters other than asterisks (*) must be provided in a term
- A minimum of 2 characters other than question marks (?) and asterisks (*) must be provided in a term

If the user knows only 2 characters of a license plate, they might not be able to use a wildcard search:

- If the user knows the position of the 2 characters, they can use the question mark wildcard character in the search. If the 2 characters are in positions 2 and 3, then the query `"?AB*"` is valid in the configuration that is described.
- If the position of the characters is not known, the user might want to search for `"*AB*"`, which is invalid in the configuration that is described.

To change the minimum number of characters that must be included in a search query with a wildcard character, edit properties in `SearchSettings.properties`. The properties that specify the minimum number of characters are:

`WildcardMinCharsWithAsterisk`

The minimum number of characters other than asterisks (*) that must be included in a wildcard query that contains an asterisk.

`WildcardMinCharsWithQuestionMark`

The minimum number of characters other than question marks (?) and asterisks (*) that must be included in a wildcard query that contains a question mark. This value should be less than, or equal to the value of the `WildcardMinCharsWithAsterisk` property.

Procedure

1. Using a text editor, open the `SearchSettings.properties` file. You can find this file in the following location: `toolkit\configuration\fragments\common\WEB-INF\classes`.
2. Edit the values of the `WildcardMinCharsWithAsterisk` and `WildcardMinCharsWithQuestionMark` properties for the Analysis Repository.
3. Save and close the file.

What to do next

After you redeploy i2 Analyze, run a selection of wildcard queries. Continue to change the values of `WildcardMinCharsWithAsterisk` and `WildcardMinCharsWithQuestionMark` until you are satisfied.

Configuring the i2 Analyze Onyx search

Translation files are used by Enterprise Insight Analysis to convert between Onyx search requests for the Information Store, and the Cognos reports format. You can generate translation files with the item types that are present in the schema that you are using.

Before you begin

This task assumes that you have IBM i2 Enterprise Insight Analysis installed, and that you have deployed a system that uses the i2 Analyze Onyx services for the Information Store.

About this task

In a system that uses the i2 Analyze Onyx services to connect to the Information Store, the translation files that map between i2 Analyze search requests and Cognos reports need to be updated if you update either the schema, or the reports. Enterprise Insight Analysis includes a tool to create translation file templates that are tailored to your schema. The template generator allows you to specify up to three files as inputs:

Schema

A complete description of all the entity types, link types, and their associated property types that are available for items within a system. In order to generate a translation file, you must provide the schema.

A list of item types to ignore

A comma separated list of the item type IDs that are present in the schema but are not used in any of the reports.

Existing ELP translation

If an existing translation tool is specified, any existing translations that relate to item types that are specified in the schema are moved to the new file, with any new item types appended ready for you to configure them.

Using the file above, 2 templates are produced:

ELP translation

This file has a entry for each of the item types that was specified in the schema. Each of these entries needs to be updated with the Cognos report information, to allow the translation to be performed.

Report translation

This file is used to handle source information and grading information to apply to items when they are returned in search results.

Procedure

1. Using a text editor, open the `toolkit\configuration\fragments\cognos-connector\WEB-INF\classes\TemplateGenerationTools.properties`.
2. Populate the value of the `SchemaFilePath` property with the full path, including the file name, to the i2 Analyze schema file for which you want to generate a translation file.
3. Set the value of the `ELPTranslationFilePath` property to the full path, including the file name, for the translation file to generate.
4. Run the setup script to generate the translation file:
   a) On the i2 Analyze server, open a command prompt and navigate to the `toolkit\scripts` directory.
   b) Run the following command:

```
setup -t generateTranslation
```

Results
You have generated the translation files using the information stored in your schema.

What to do next
Before the translation files can be used, you must update the file with the Cognos report information that matches your deployment. See"Updating the translation files to match your reports" on page 140.

Updating the translation files to match your reports

Each time that you generate a translation file, or change the reports that you would like to run, you need to populate your translation file with the information it needs to identify items in Cognos. By providing this information, you align the item types that are stored within the schema with the Cognos objects that they represent, allowing i2 Analyze searches to return results.

About this task
When you generate translation files from the i2 Analyze schema:

- An entry for each item type is added to the item translation file allowing you to map these to Cognos objects.
- A list of the reports that relate to these item types.
- A list of the available search filters is created in the item translation file.
- A separate results translation file is produced allowing you to specify source and grading information.

Although default values are provided to identify both the Cognos objects and the corresponding reports, these will need to be modified to match the actual values that are present in your deployment of Cognos.

Procedure

1. Using an XML editor, open the item translation file that you generated.
2. For each `ItemType` element:
   a) Remove the element if you do not want to search on this item type.
   b) Ensure that the `CognosItemType` attribute is set to the correct identifier for your system.

c) Remove any `PropertyType` elements that are associated with property types you don't want to search for.

d) For all geospatial property types, you must specify the following additional attributes:

Attribute	Description
`GeospatialId`	The type of geospatial search that can be carried out on properties of this type. By default, the available values are `CDR` (call detail record) and `NLD` (network location data). These options, and any others you add, must correspond to the geospatial search types that are configured for your deployment. **Note:** When you specify the `GeospatialId` attribute, you must also include the `VisualQueryChain` attribute.
`VisualQueryChain`	A comma-separated list that represents the series of item types in the query. This must specify either a single entity or a linear chain of alternating entities and links, starting and ending with an entity. You must include at least one instance of the item type that is specified in the parent `ItemType` element.

3. For each `ReportCollection` element:

   a) Remove the element if you do not want to define a search of this type.

   b) Ensure the `xsi:type` attribute is set to the type of report:

   - `ns3:EntityReports` - A report about a specific entity type
   - `ns3:DumbellReports` - A report about a pair of linked entity types
   - `ns3:ExpandReports` - The supplied analytic that allows linked entities to be found for a specified entity
   - `ns3:FindPathReports` - The supplied analytic that searches for connections between two entities

   c) In the `SummaryReport` Element enter the `Name` of the report.

   d) Depending on the type of search you selected, populate the remainder of the `ReportCollection` attributes:

   - `ItemTypeId` - Identifies a single item type
   - `End1TypeId` - Identifies a valid entity type for the first end of a link
   - `LinkTypeId` - Identifies a valid link type for the specified link ends
   - `End2TypeId` - Identifies a valid entity type for the second end of a link
   - `End2TypeCognosPromptName` - Allows you to specify the Cognos prompt to use to select the valid values for the second end.

4. In the `SupportedFilterOperators` section, review the available search filters, and remove any that you do not wish to make available.
5. Save and Close the file to store your changes.
6. Optional: If your system is set up to require mandatory source information or grading, add the details to your report results file.

Validating translation files

Once you have created a translation file and updated the contents to match your reports, you can validate the files before attempting a full deployment. Validating the files early allows you to detect and resolve any issues.

Procedure

Run the setup script to validate the translation file:

a) On the i2 Analyze server, open a command prompt and navigate to the `toolkit\scripts` directory.

b) Run the following command:

```
setup -t validateTranslation
```

Results

The files provided will be checked to ensure that the structure of the xml is valid, and that all mandatory elements and attributes have been specified.

What to do next

Modify the deployment toolkit to specify that i2 Analyze uses the new translation files, and then redeploy the platform and conduct your tests. If necessary, iterate over these steps again until you have the translation files that you need.

Understanding translation files

Every deployment of IBM i2 Enterprise Insight Analysis that uses the Onyx services to access data in the Information Store must include an item translation file. The contents of the file depend on the schema and on the reports that are available in your deployment.

In outline, the XML file that translates item information into a format that can be used by reports has the following structure:

```
<ArgumentTranslation>
    <ItemTypes>
        <ItemType>
            <PropertyTypes>
                <PropertyType/>
                ...
            </PropertyTypes>
        </ItemType>
        ...
    </ItemTypes>
    <ReportCollections>
        <ReportCollection>
            <SummaryReport/>
        </ReportCollection>
        ...
    </ReportCollections>
    <SupportedFilterOperators>
        <FilterOperator/>
        ...
    </SupportedFilterOperators>
</ArgumentTranslation>
```

Inside the `<ArgumentTranslation>` root element, the XML file has three main sections:

- `ItemTypes` contains an entry for each available item type. Each entry contains a list of the property types that the reports make available for users to specify in searches.
- `ReportCollections` enumerates the different reports that can return information for each item type that is identified in the first section. When a link type supports different combinations of entity types at the ends of the link, you must define a report collection for every combination.

  The item and property types that are associated with a report collection define which item and property types are available in visual query. For each item, the report collection specifies the analytic that is used for each link type and end combination.
- `SupportedFilterOperators` lists the operators that users can specify against items when they create a visual query. Values in this list must match constants from the `FilterOperator` enumeration. For more information, see the IBM i2 Analyze API

Item types

The first section of the translation file defines the item and property types that can be included in a visual query. The list of item types in the translation file can be a subset of the types that are available in the schema.

In the translation file, the item types that the deployment supports are described inside a parent `<ItemTypes>` element. Each `<ItemType>` child element defines a translation from an i2 Analyze item

type to a specific Cognos item type. In turn, the <PropertyType> child elements list the property types that are associated with the parent item type.

Item types

For an item of a particular type in the i2 Analyze schema, a search can produces results that contain the corresponding Cognos item type.

Every <ItemType> element has two mandatory attributes:

Attribute	Description
Id	The identifier in the i2 Analyze schema for the specified item type.
CognosItemType	The internal database item type. This value is the Cognos item type.

Property types

Inside an <ItemType> element, the mandatory <PropertyTypes> child element contains a list of the available property types for that item type.

Note: Only those property types that you want to make available for users to constrain and filter their searches must be specified.

If an item type does not have any properties for which users can search, then the <PropertyTypes> element can be left empty. Otherwise, it contains <PropertyType> child elements.

Every <PropertyType> element is identified using the schema Id and can have more attributes:

Attribute	Description
Id	The identifier in the schema for the specified property type. This attribute is mandatory.
Name	A name that describes the property type. This optional attribute is provided to help you identify the property types more easily.
GeospatialId	An attribute that is only available for Indicates that users can search for items with property values that are in a defined geospatial area, rather than matching a specific set of coordinates. This value is a three letter code that allows geospatial properties to be grouped within the user interface. By default, the available values are CDR (call detail record) and NLD (network location data). These options and any others you add must match values specified in the EIAEsriSearchCommands list in ApolloEIAClientSettings.xml. **Note:** You can enable geospatial searches for only one item type per Cognos item type, and when you specify the GeospatialId attribute, you must also include the VisualQueryChain attribute.
VisualQueryChain	A comma-separated list that represents the series of item types in the query that is created when a user defines a geospatial search area. You use the identifiers in the schema to identify the item types. The query structure that you define must be either a single entity or a linear chain of alternating entities and links, starting and ending with an entity. **Note:** You must include in the query at least one instance of the item type that is specified in the parent <ItemType> element.

Report collections

In the translation file, the <ReportCollections> element contains information that enables Enterprise Insight Analysis to select and run the appropriate report. Report collections define the reports that can be run. The choice of which report to run depends on the type and structure of the results that the user requires.

When a user searches the Information Store, Enterprise Insight Analysis responds by running a Cognos report.

The contents of the <ReportCollections> element reflect the different ways that users can request data. The following report types are included:

Report Type	Description
Entity	Searches for information that is specific to a set item type.
Dumbbell	Searches for information that is specific to a pair of linked entity types.
Expand	Uses the expand analytic to search for entities that are linked to a selection.
Find Path	Uses the find path analytic to search for connections between entities.

The translation file contains a <ReportCollection> element for each of the structures that an Enterprise Insight Analysis user can request.

Report collections for entity searches

When a user submits a search request, Enterprise Insight Analysis initially runs a summary report.

The translation file contains one <ReportCollection> element for each entity type that can appear in the results. For example:

```
<ReportCollection xsi:type="ns3:EntityReports" ItemTypeId="ET3">
    <SummaryReport Name="VQ_Entity_Vehicle_sp1"/>
</ReportCollection>
```

The xsi:type attribute indicates the search structure for which the reports are intended. The ItemTypeId attribute specifies the item type identifier from the item type definition in the translation file.

Each <SummaryReport> child element has a Name attribute that identifies the Cognos report. When it is appropriate to do so, you can use the same report for different purposes.

Report collections for dumbell searches

The requirements on reports that run in response to searches for links are similar to the requirements on reports for entity searches. In the translation file, the <ReportCollection> elements for a link type

contain more information than for a single entity type. The extra content is required to identify the entity types that are connected by links of a particular type in the Information Store. For example:

```
<ReportCollection xsi:type="ns3:DumbbellReports" End1TypeId="ET5"
  LinkTypeId="LT4" End2TypeId="ET4"
  End2TypeCognosPromptName="p_pVal1_toEntityType">
    <SummaryReport Name="VQ_Link_MemberOf_sp1"/>
</ReportCollection>
```

In comparison to the `<ReportCollection>` element for reports that return entities, the `<ReportCollection>` element for link reports has different features. The element itself contains extra attributes that precisely specify the items that the reports return:

Attribute	Description
`End1TypeId`	The identifier from the i2 Analyze schema for the entity type at the "from" end of the link in the structure for which the report collection is intended.
`LinkTypeId`	The identifier from the i2 Analyze schema for the link type in the structure for which the report collection is intended.
`End2TypeId`	The identifier from the i2 Analyze schema for the entity type at the "to" end of the link in the structure for which the report collection is intended.
`End2TypeCognosPromptName`	This value is used by analytics for conditional rendering of page objects. For example, if a link type can have more than one end type, the value for `End2TypeCognosPromptName` specifies the filter that is used to visualize the data for the specified end type.

Supported filter operators

The third section of the XML file that translates requests from i2 Analyze into a form that is compatible with Cognos is a list of operators for search conditions. The presence of an operator in the list determines whether users can specify that operator when they search the Information Store.

The `<SupportedFilterOperators>` element has no attributes and the only child elements are `<FilterOperator>` elements that take values from the `FilterOperator` enumeration in the i2 Analyze API. In the translation file for most Enterprise Insight Analysis deployments, the `<SupportedFilterOperators>` element is similar to the following example:

```
<SupportedFilterOperators>
    <FilterOperator>STARTS_WITH</FilterOperator>
    <FilterOperator>EQUAL_TO</FilterOperator>
    <FilterOperator>GREATER_THAN</FilterOperator>
    <FilterOperator>GREATER_THAN_OR_EQUAL_TO</FilterOperator>
    <FilterOperator>LESS_THAN</FilterOperator>
    <FilterOperator>LESS_THAN_OR_EQUAL_TO</FilterOperator>
    <FilterOperator>BETWEEN</FilterOperator>
</SupportedFilterOperators>
```

Although the "smart match" operator is included in the `FilterOperator` enumeration in the API, Enterprise Insight Analysis does not support this operator. If you remove a `<FilterOperator>` element,

then users cannot select that operator in the conditions for all searches of the Information Store. You cannot enable and disable specific operators for individual item types.

Report results
The report results translation file is an optional file that can contain grading information and identifying source information. If this information is supplied, all items displayed in the Intelligence Portal from the Information Store will have the same values for this information.

Grade information

The <Grades> element specifies the grades that users can assign to properties. Each <Grade> child element has attributes that set the identifier and value of a grade to be assigned:

```
<Grades>
   <Grade GradeTypeId="..." Value="..."/>
   ...
</Grades>
```

For example:

```
<Grades>
   <Grade GradeTypeId="G1" Value="A"/>
   <Grade GradeTypeId="G2" Value="1"/>
   <Grade GradeTypeId="G3" Value="1"/>
</Grades>
```

Note: If the schema defines mandatory grades, then you must ensure that those grades receive values in the translation file.

Source information

The <SourceInfo> element specifies the source information that users can assign to cards. The <SourceInfo> element has attributes that set the type and reference values and contains a <Description> child element to allow a larger description to be assigned:

```
<SourceInfo Type="" Reference="">
   <Description>
   </Description>

</SourceInfo>
```

For example:

```
<SourceInfo Type="Information Store" Reference="IS1">
   <Description>This information was imported from the Information Store.
   </Description>

</SourceInfo>
```

Moving a deployment toolkit configuration

You can move the configuration directory between installations of the same version of the deployment toolkit. When you move your configuration to a new environment, you must update that configuration with the details of the new system.

About this task

Before you can move a deployment toolkit configuration to a new environment, you must ensure that the same version of the deployment toolkit is installed in both the source and destination environment. For more information about how to install the deployment toolkit, see Installing IBM i2 Analyze.

Important: If you are copying into a destination environment that is already configured, make a copy of the existing configuration files before you replace the contents.

Important: Any configuration changes that you completed outside of the `configuration` directory must be completed again in the new environment.

Procedure

1. In the source deployment, navigate to the `toolkit` directory.
2. Copy the `configuration` directory to the `toolkit` directory of the target i2 Analyze installation.
3. Update the values in the configuration files to match the new environment.

   For more information about modifying the configuration files, see “Modifying the basic deployment” on page 1.
4. Deploy i2 Analyze on the new system.

   For more information, see “Deploying i2 Analyze” on page 150.

What to do next

Test that the deployment works correctly in the new environment.

Redeploying and resetting i2 Analyze

Any time that you change the configuration of i2 Analyze, you must redeploy the system. During development, it is common also to need to clear data from the i2 data stores. The i2 Analyze deployment toolkit includes commands that perform both of these operations.

About this task

Significant changes to the schema or the security schema of an i2 Analyze deployment usually invalidate the information in i2 data stores. As you develop your schemas, you often need to remove and re-import your test data.

After any change to the toolkit configuration of a development or production deployment of i2 Analyze, you must redeploy in order to push the change to the running deployment.

Performing these actions in sequence is common before your deployment goes live, but rare afterward. If you do run them in sequence, the combination effectively resets the system to its just-deployed state.

Clearing data from the system

The `clearData` task can be used to remove data that is stored in both the index and the database. You can use this task to remove test data from a system.

About this task

Important: These instructions are designed to remove data that is used for test purposes. Do not use the task to clear data in a production system without backing up your data. For more information about backing up your data, see “Backing up a deployment” on page 151.

Procedure

1. On the i2 Analyze server, open a command prompt and navigate to `toolkit\scripts`.
2. To clear data and the search index, run the following command:

```
setup -t clearData
```

The following message is displayed when you run the `clearData` task:

```
Are you sure you want to run the 'clearData' task?
This will permanently remove data from the system. (y/n)
```

Enter `Y` to continue. The data and the search index are removed from the system.

Removing databases from the system

The `dropDatabases` task can be used to remove not only the data, but also the underlying structures such as tables from your database management system. You can use this task to remove existing databases before you recreate the databases in a different location.

About this task

Important: These instructions are designed to remove databases that are used for test purposes. Do not use the task to remove databases in a production system without backing up your data. For more information about backing up your data, see “Backing up a deployment” on page 151.

Procedure

1. On the i2 Analyze server, open a command prompt and navigate to `toolkit\scripts`.
2. To clear data and the search index, run the following command:

```
setup -t dropDatabases
```

The following message is displayed when you run the `dropDatabases` task:

```
Are you sure you want to run the 'dropDatabases' task?
This will permanently remove data from the system. (y/n)
```

Enter `Y` to continue. The data and the search index are removed from the system. The next time that you deploy the system, a new database is created.

Deploying i2 Analyze

To ensure that your configuration changes are included, you need to deploy them. You might need to deploy multiple times whilst building up your configuration.

About this task

The first time that you deploy i2 Analyze on a server, the script run through all the available deployment actions. When you redeploy i2 Analyze, the script updates only the areas of the deployment that need modification. This behavior reduces the time that it takes for subsequent deployments.

If you have updated the configuration files with the correct information for your deployment, then the `setup` script performs the following actions, depending on whether the areas of the deployment are modified or not:

- Set up your DB2 data stores
- Configure IBM HTTP Server to act as a reverse proxy
- Create and configure the application server profiles
- Create and deploy the specified applications

Note: If you have a system that includes multiple data sources, generally these have been set up to use different server profiles. There are multiple tasks that are able to function in an environment with a single profile that need to identify the correct profile in situations where there are multiple. To specify the server profile, you must add the `-s` flag with the name. For example, to start the server profile for the Information Store, run the following command:

```
setup -t start -s opal-server
```

To start the server with the Analysis Repository, run the following command:

```
setup -t start -s onyx-server
```

Procedure

1. Open a command prompt and navigate to `toolkit\scripts`.
2. Ensure that the application servers are stopped. To stop the application server, run the following command:

   ```
   setup -t stop
   ```
3. To deploy i2 Analyze components, run the following command:

   ```
   setup -t deploy
   ```

   Use the information in Troubleshooting the deployment process to ensure that i2 Analyze is deployed successfully.
4. To start the application server, open a command prompt and navigate to `toolkit\scripts`. Then, run the following command:

   ```
   setup -t start
   ```
5. Start, or restart, the HTTP server that hosts the reverse proxy.

Backing up a deployment

An effective backup strategy considers the key components in your deployment and to assess the impact of creating replica versions versus the cost of replacing those components if the system fails. All live deployments of i2 Analyze should have a backup strategy in place.

The items that you back up can include not only databases, but also search indexes and the customized files that are used to deploy the system.

When you create a backup strategy, it is important to consider the impact of data loss. Other considerations include:

- The amount of time, if any, that is acceptable for a system to be offline to create backups.
- The amount of time that is acceptable for a system to be offline while recovery is in progress.
- The amount of data, if any, that is acceptable to be unrecoverable by the system.

Backing up an offline i2 Analyze system

An offline back up is taken when the system is not in use. The advantage in taking a backup when the system is not in use is that you have confidence that the

About this task

For a deployment that uses the Analysis Repository, back up the Analysis Repository database, and the Lucene index directory.

For a deployment that uses the Information Store, back up the Information Store database, and the Solr and ZooKeeper directories.

Regardless of which data stores that you are using, your backup procedure might also include backing up the following items:

- The data directory for i2 Analyze.
- The i2 Analyze deployment toolkit configuration files.
- The home directory for WebSphere Application Server Liberty profile.
- The configuration files for the HTTP reverse proxy.

The `environment.properties` files in the configuration contain the locations for many of these items. Each application that is defined in your topology file has an associated folder in the `toolkit\configuration\environment\` directory that stores information about the environment that is specific to that application.

Procedure

1. Ensure that any application servers are stopped.

   To stop the application server for the Analysis Repository, run the following command:

   ```
   setup -s onyx-server -t stop
   ```

   To stop the application server for the Information Store, run the following command:

   ```
   setup -s opal-server -t stop
   ```

2. Create a backup of the databases in your system by following the instructions for your database management system.

Note: If you are using IBM DB2 as your database management system, the root location of DB2 database file storage is specified by the `db.database.location.dir.db2` property in the `environment.properties` file.

3. Make a copy of the data directory for i2 Analyze.

   The default directory is `C:\IBM\i2analyze\data`. The data directory is specified by the `apollo.data` property in the `environment.properties` file.

   By default the Lucene index, Solr, and ZooKeeper directories are located within this directory.

4. Optional: If your Lucene index, Solr, and ZooKeeper directories are not in the default locations, make a copy of the directories.

   The directory locations are specified in the `topology.xml` file, which is stored in the `toolkit\configuration\environment` directory.

   The following elements in the `topology.xml` file contain the directory locations:

   - For the Lucene index directories, the locations of each index are specified in a `<lucene-index>` element as the values of the `main-index-location` and `alternatives-location` attributes.
   - For the Solr directories, the locations are specified in the `<solr-node>` element as the value of the `data-dir` attribute.
   - For the ZooKeeper directories, the locations are specified in the `<zkhost>` element as the value of the `data-dir` attribute.

5. Make a copy of the i2 Analyze deployment toolkit directory that contains the configuration files. The default directory is `toolkit\configuration\`.
6. Make a copy of the home directory for WebSphere Application Server Liberty profile.

   The default directory is `C:\IBM\i2analyze\deploy\wlp\usr`.

   The directory name is specified in the `wlp.home.dir` property in the `environment.properties` file.

7. Make copies of the HTTP reverse proxy configuration files.

   Copy the `httpd.conf` file that is stored in the `C:\IBM\HTTPServer\conf` directory, and the `plugin-cfg.xml` file that is stored in the `C:\IBM\HTTPServer\plugins\iap\config` directory.

8. When your backups are complete, restart the application servers for your system.

   To start the onyx application server, run the following command:

   ```
   setup -s onyx-server -t start
   ```

   To start the opal application server, run the following command:

   ```
   setup -s opal-server -t start
   ```

Recovering an offline backup

To recover a backup that is taken from an offline system, you must restore the databases in your system and copy the files that you backed up into their original locations.

Procedure

1. Ensure that any application servers are stopped.

   To stop the application server for the Analysis Repository, run the following command:

   ```
   setup -s onyx-server -t stop
   ```

To stop the application server for the Information Store, run the following command:

```
setup -s opal-server -t stop
```

2. Restore the databases in your system by following the instructions for your database management system.
3. Copy the backups of the following items into their original locations:
   - The data directory for i2 Analyze.
   - The Lucene index directories for the Analysis Repository.
   - The Solr and ZooKeeper directories for the Information Store.
   - The i2 Analyze deployment toolkit configuration files.
   - The home directory for WebSphere Application Server Liberty profile.
   - The configuration files for the HTTP reverse proxy.
4. Start the application servers for your system.

   To start the application server for the Analysis Repository, run the following command:

   ```
   setup -s onyx-server -t start
   ```

   To start the application server for the Information Store, run the following command:

   ```
   setup -s opal-server -t start
   ```
5. Start, or restart, the HTTP server that hosts the reverse proxy.

Results

After the application servers start, the system is ready for use.

Backing up an online i2 Analyze system

When the system is in use, you can back up the Analysis Repository and Information Store databases. When you restore the databases, you must reindex the data in the data store.

Procedure

Create a backup of the databases in your system by following the instructions for your database management system.

By default, the Analysis Repository database is named *WRITESTO* and the Information Store database is named *ISTORE*.

Recovering an online backup

To recover a backup that is taken from an online system, you must restore the databases in your system. After you restore the databases, you must reindex the data in the data store.

Note: Depending on the amount of data in the store, and the specification of the server, the reindex might take a long time to complete.

Procedure

1. Ensure that any application servers are stopped.

To stop the application server for the Analysis Repository, run the following command:

```
setup -s onyx-server -t stop
```

To stop the application server for the Information Store, run the following command:

```
setup -s opal-server -t stop
```

2. Restore the databases in your system by following the instructions for your database management system.
3. Clear the search index from the system.

   To clear the search index for the Analysis Repository, run the following command:

   ```
   setup -s onyx-server -t clearSearchIndex
   ```

   To clear the search index for the Information Store, run the following command:

   ```
   setup -s opal-server -t clearSearchIndex
   ```

4. Start the application servers for your system.

   To start the application server for the Analysis Repository, run the following command:

   ```
   setup -s onyx-server -t start
   ```

   To start the application server for the Information Store, run the following command:

   ```
   setup -s opal-server -t start
   ```

   The reindex process for each application server begins when you start the application server. The following message is displayed in the console: `# Starting search index rebuild.`

   Note: If the reindex process is interrupted, the process continues from where it was interrupted the next time that you start the application server.
5. Start, or restart, the HTTP server that hosts the reverse proxy.

Results
After the reindex process is complete, the system is ready for use.

Setting up a localized deployment

i2 Analyze is translated into a number of languages. To deploy i2 Analyze in a different language, a number of customizations need to be made.

When you deploy i2 Analyze, it is possible to display the following localized components:

Intelligence Portal

The Intelligence Portal contains the i2 Analyze strings that are displayed to users. By default, if the language of the operating system is supported, the Intelligence Portal displays in that language. You can change the language of the Intelligence Portal by adding the following parameter to the end of your url:

```
?lang=xx-XX
```

Where `xx-XX` is one of the supported language codes. For example:

```
http:/my_server:8080/apollo/?lang=fr-FR
```

Schema

Schemas contain information about the item types and their associated property types. For each of the example schemas that is included with i2 Analyze, localized versions are also included. The example schemas can be used directly, or modified to suit the needs of your deployment.

Note: As the schema is stored within the Analysis Repository, only one language is supported for each deployment. As such, if the Intelligence Portal strings are changed to a different language, the item type, and property type labels remain in the originally specified language.

Search index

When data is added to the system, the platform recognizes terms based on files that have been provided in a specific language. Although direct matching is supported for data that is provided in other languages, smart matching and synonyms are only provided for that specified language.

In addition, you can set the Intelligence Portal to display text in a left-to-right mode, or a right-to-left mode, or to automatically detect the display direction based on the context.

Enabling bi-directional text support

For certain languages, such as Hebrew and Arabic, text is intended to be displayed right to left. You can configure the behavior of the Intelligence Portal, to determine the direction in which to display text.

About this task

The `ApolloClientSettings.xml` file contains two settings that you can configure to enable bi-directional support in i2 Analyze.

BidiEnabled

Whether to enable bidirectional support for strings that the system provides. If empty, false, or invalid, bidirectional support for strings is not enabled.

TextDirection

Whether to enable bidirectional support for strings that users provide. Possible values: 0: Left-to-right only 1: Right-to-left only 2: Contextual (determined by the first strong directionality character)

Procedure

1. Using a text editor, open the `ApolloClientSettings.xml` file. You can find this file in the following location: `toolkit\configuration\fragments\onyx-services-ar`.
2. Using the preceding descriptions, populate the property values for the `BidiEnabled` and `TextDirection` properties.
3. Save and close the file.

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 grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Limited Hursley House Hursley Park Winchester, Hants, SO21 2JN UK

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other names may be trademarks of their respective owners. Other company, product, and service names may be trademarks or service marks of others.