Hewlett Packard
Enterprise

Developer's Guide

HPE ArcSight Actor Model Import FlexConnector for Database 7.0.7.7289

January 16, 2015

Contact Information

Phone	A list of phone numbers for HPE ArcSight Technical Support is available on the HPE Enterprise Security contacts page: www.hpe.com/software/support/contact_list
Support Web Site	www.hpe.com/software/support
Protect 724 Community	https://www.protect724.hpe.com

Contents

Chapter 1
Overview

The Actor Model Import FlexConnector for Database provides a way to read and import identity information from SQL databases into the ArcSight ESM v5.0 SP1 and above actor model. The connector uses database parsers to read and import the required identity information from one or more tables. The connector supports both time-based and ID-based database tables.

You configure the connector using the SmartConnector Configuration Wizard. You can configure three types of parsers:

- **Base attributes** - these are the attributes associated with a user and correspond to the ESM actor model attributes.
- **Account attributes** - these are the user account attributes that are loaded into ESM.
- **Role attributes** - these are the user role attributes that are loaded into ESM.

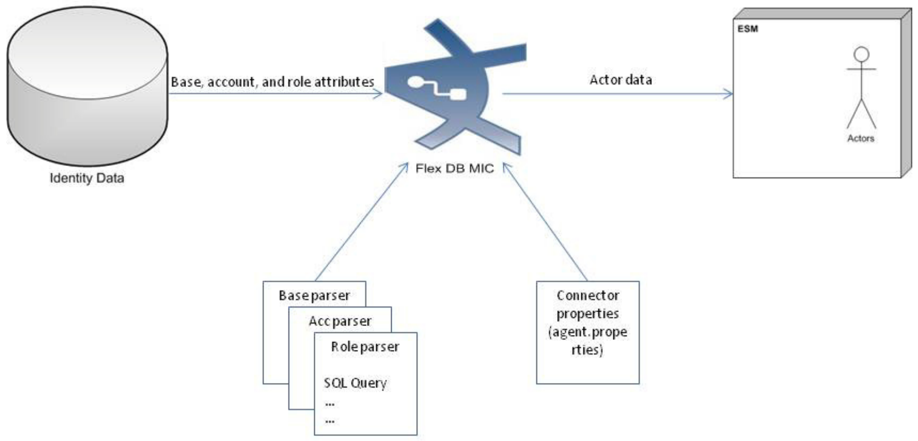
The connector supports two modes of operation:

- Initial read and import
- Ongoing detection and import of updates

During the intial read and import for base attributes, the connector can import a full set of users and a specified set of attributes for each user or the connector can import a specified subset of users and attributes based on the parser query. The connector can also do an initial read and import of roles, authenticators and accounts.

During detection and import of updates, the connector will check for updates at the interval specified during connector configuration. The connector is dependent on row updates to detect changes. Some changes to the database might not provide enough information to the query to detect the changes. For example, deleting a user might not generate a new database record. In that case, you might need to create a new table, trigger, view or a combination of these depending on your environment so that the parser query can detect the change.

Figure 1-1 Actor Model Import FlexConnector for Database Component View



Assumptions

- To successfully implement the connector, you should have experience with databases and SQL.
- You should be familiar with writing flex database parsers. Refer to the *FlexConnector Developer's Guide* for more information about writing a database parser.
- The UUID or unique id should be unique across all sources if data from multiple sources is being sent to a single ESM destination.
- The IDM Identifier should remain the same between initial import and ongoing updates. If the IDM Identifier changes after the initial import, ESM will not be able to relate ongoing updates to the right actor.

Chapter 2

Actor Model Import FlexConnector for Database

This chapter provides information about the Actor Model Import FlexConnector for Database.

The following topics are covered here:

Actor Attributes

The Actor Model in ESM consists of three types:

- Base attributes
- Account attributes
- Role Attributes

Extracting the information from your identity management system (IMS) or database tables containing identity information is primary in setting up the Actor Model Import FlexConnector for Database. The following sections provide the attributes in ESM for the three types of information to be extracted.

Base Attributes

The base attributes for the Actor Model in ESM are what you map to when you configure the connector and write queries to extract identity information from your identity management system or your database containing identity information. Depending on your IMS or database tables containing identity information, all or some of the attributes might apply. You might want to map to only a subset of the attributes depending on your use case.

The base attributes for the Actor Model in ESM are:

Attribute	Description
UUID	**Required** The Universally Unique Identifier for the actor. This is the alphanumeric strong name generated by the IDM to identify this user. The UUID should be unique across all sources if data from multiple sources is being sent to a single ESM destination.
Full Name	**Required** The actor’s full name as concatenated in the IDM or database.
First Name	Specifies the actor’s first name.
Last Name	Specifies the actor’s last name.
Middle Initial	Specifies the actor’s middle initial.
IDM Identifier	String used to identify the IDM in the ESM console and group identities under one group named after the IDM Identifier. Once the IDM Identifier is set, you cannot change it without re-configuring the connector.
DN	The distinguished name for the user, for example, CN=John Doe, OU=Sales, DC=companyname,DC=com
Employee Type	The type of employee this actor is in your company. This value is usually a classification unique to your company’s personnel operations, for example, full-time, exempt, or contractor.
Status	The employment status of the actor, one of: Active, Deleted or Disabled. When an actor is deleted from the IDM or database, the actor will remain in the ESM actor model with the status of deleted. This will preserve any history related to this actor in case activity appears on the system that is inappropriate to the actor’s status. If the actor is deleted directly from ESM, the actor will be completely removed from the ESM actor model without preserving a history.
Title	Specifies the actor’s job title.
Company	The company by whom the actor is employed, applies to contractors or employees from partner companies.
Org	The organization within your company of which the actor is a member.
Department	The actor’s department.
Manager	The actor’s manager.
Assistant	The actor’s assistant.
Email address	The actor’s company email address.
Location	The actor’s work location.
Office	The actor’s office address.
Business Phone	The actor’s business phone.
Cell Phone	The actor’s mobile phone.
Fax	The actor’s fax number.
Pager	The actor’s pager number.

Attribute	Description
Address	Actor’s business street address.
City	Actor’s business address city.
State	Actor’s business address state.
ZIP Code	Actor’s business address ZIP code.
Country or Region	Actor’s business address country or region.

Account Attributes

The account attributes for the Actor Model in ESM provide the accounts information that is correlated to each actor. When configuring the connector, these attributes are the information you want to extract from your IMS or database tables containing account information. The account attributes are:

Attribute	Description
Account Name	**Required** - This is attribute contains all the user’s account IDs tracked in the authentication data store, for example, john_doe, jdoe, or john.d. It is required.
Authenticator	**Required** - This is the friendly name for the user authentication data store derived by the connector based on information in the IMS or database, for example: “Active Directory:mycompany.com”, “Oracle”.

Role Attributes

The role attributes for the Actor Model in ESM provide the various role information that is associated with an actor. When configuring the connector, these attributes are what you seek to extract from your IMS or database tables containing role information.

Attribute	Description
Role Name	**Required** - Name of the role, such as Manager or Administrator.
Resource Type	Name of the application, organization, or network resource for which that person performs the role, such as the name of your company, the name of the application to which the user has privileges, or the name of a network device to which the user has privileges.
Role Type	The role type is the role’s category. For example, Global Security Group or Local Distribution Group.

SQL Queries

The SQL queries that you write in the parsers serve two purposes: the initial import of attributes into ESM and updates of the attributes.

Initial Import

The connector is ready for initial import, once it is configured and the parsers are completed as explained in previous sections. When the connector runs for the first time, it imports all the actor data, as specified in the queries, into ESM and imports any new actors added there onward as an ongoing update. During the initial import of attributes into ESM, ensure that all the attributes that you are interested in tracking are included in the query. Once actor information is imported into ESM, the list of attributes the connector sends to ESM for existing actors is not updated. If you add or remove attributes to be sent to ESM from the connector after you import the actor model, you will not get a history of the new attributes. Updates will only be from the point of time the attributes were added. If you want a history of the added attributes, re-import the actors.

Updates

The SQL queries can be either time or ID based depending on how the data is structured. The queries should be able to find the delta of the changes. If the data structure is such that the queries are not able to find the attributes and the delta of them, you might need to use some techniques like joining tables, creating tables, creating views, creating triggers and so forth.

After initial import, the connector should send the data to ESM as a delta (changes only). The UUID is always required. When sending the delta, ensure that the time of the change is included as StartTime. If a StartTime is not provided, current system time is used. Write the parser in such a way that when the query is executed after the initial import it finds only the changes. Refer to the *FlexConnector Developer's Guide* for more information about writing a database parser.

Ongoing updates can be divided into base, account, and role parsers. To terminate the attributes:

- **Base attributes:** the actor can be disabled or marked deleted by sending the appropriate value in the Status field.
- **Account attributes:** accounts can be terminated by sending `-9223372036854775808` as the StartTime and termination time as the EndTime.
- **Role attributes:** roles can be terminated the same way as account attributes.

Chapter 3

Installing and Configuring the Connector

This chapter provides information about the prerequisites, installation and configuration of the Actor Model Import FlexConnector for Database.

The following topics are covered:

Prerequisites

Before installing the Actor Model Import FlexConnector for Database, the following prerequisites must be met:

- Ensure that ArcSight ESM 5.0 SP1 or later and Console are installed. For more information, see the *ArcSight Installation and Configuration Guide 5.0* or later.
- Local access to the machine where the Actor Model Import FlexConnector for Database is to be installed and administrator privileges to that machine.
- A minimum of 256 MB of memory and 3 GB of available hard disk space on the host machine.
- ArcSight ESM and database components must be up and running to configure the Actor Model Import FlexConnector for Database.

Supported Platforms

The Actor Model Import FlexConnector for Database supports the following platforms:

- Microsoft Windows Server 2012 R2, 64-bit
- Microsoft Windows Server 2008 R2, 64-bit
- Microsoft Windows Server 2003 R2 SP2, 64-bit
- Red Hat Enterprise Linux (RHEL) 6.5, 64-bit

- Red Hat Enterprise Linux (RHEL) 6.2, 64-bit
- Red Hat Enterprise Linux (RHEL) 5.7, 64-bit

Installing the Actor Model Import FlexConnector for Database

This section provides instructions on how to install the Actor Model Import FlexConnector for Database.

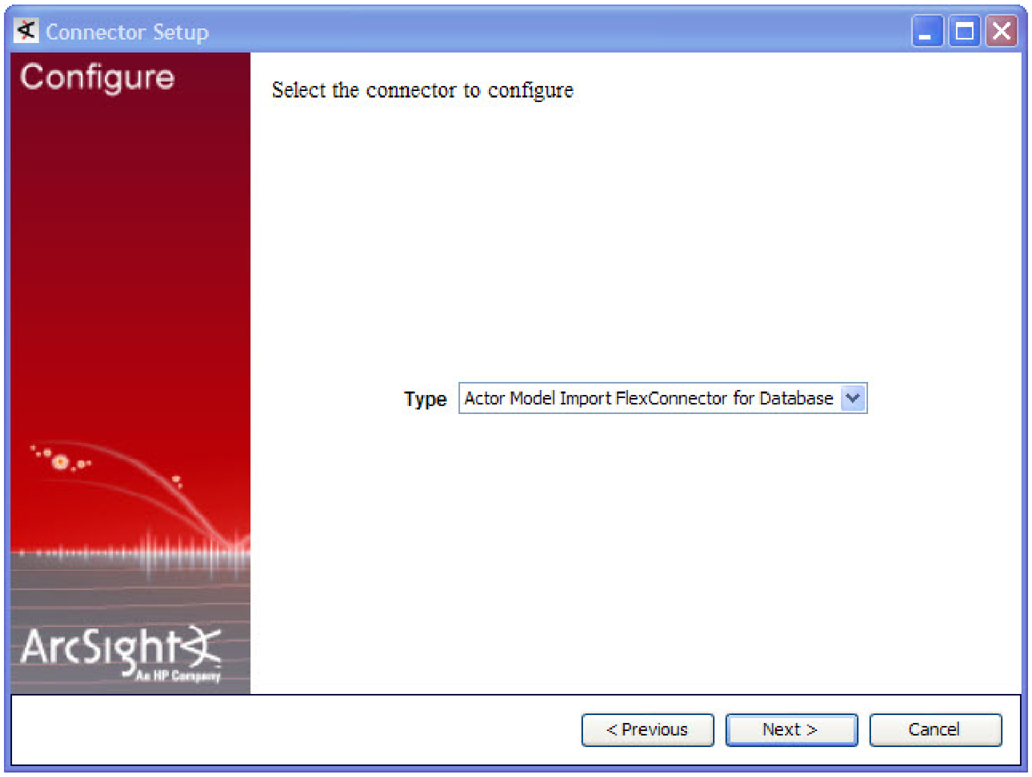
1. Using the log-in credentials supplied to you by HP, download the Actor Model Import FlexConnector for Database from the ArcSight download site (https://software.arcsight.com/) to the machine where the connector will run. The executable files for the supported platforms are:
   - `ArcSight-7.0.7.7289.0-FlexDBActorModelConnector-Win64.exe`
   - `ArcSight-7.0.7.7289.0-FlexDBActorModelConnector-Linux64.bin`
2. Place the executable file in a directory.
3. Double-click the executable file to start the installer.
4. Follow the installation wizard through the following folder selection tasks and installation of the core connector software:
   - Introduction
   - Choose Install Folder
   - Choose Shortcut Folder
   - Pre-Installation Summary
   - Installing...

Note

When selecting destinations for the Actor Model Import FlexConnector for Database, select ESM Manager only. No other destinations are supported.

Configuring the Actor Model Import FlexConnector for Database

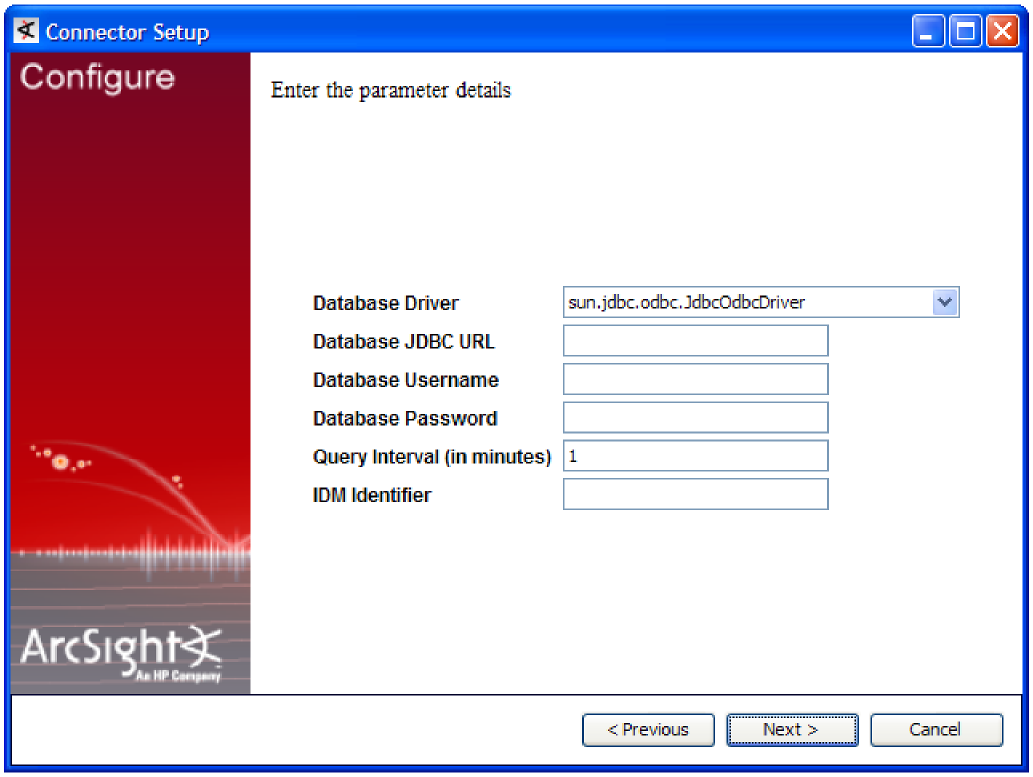
This section provides information about configuring the Actor Model Import FlexConnector for Database. After installation completes, the Configuration Wizard displays:



Select the connector to configure

Type: Actor Model Import FlexConnector for Database

Select the **Actor Model Import FlexConnector for Database** and then click **Next**. The required parameters screen displays:



Enter the parameter details

Database Driver	sun.jdbc.odbc.JdbcOdbcDriver
Database JDBC URL	
Database Username	
Database Password	
Query Interval (in minutes)	1
IDM Identifier	

Enter the values for the parameters. After entering values, click **Next**.

Parameter	Description

Database Driver	Specify the type of driver for your database.
	For Oracle, use:
	`oracle.jdbc.driver.OracleDriver`
	For MySQL, use:
	`org.gjt.mm.mysql.Driver`
	For ODBC, use:
	`sun.jdbc.odbc.JdbcOdbcDriver`
	For Microsoft SQL Server, use:
	`com.microsoft.sqlserver.jdbc.SQLServerDriver`
	For PostGreSQL, use:
	`org.postgresql.Driver`
	For DB2 unified driver, use:
	`com.ibm.db2.jcc.DB2Driver`
	For DB2 Legacy CLI-based, use:
	`COM.ibm.db2.jdbc.net.DB2Driver`
	For Sybase, use:
	`com.sybase.jdbc2.jdbc.SybDriver`
	For CsvJdbc, use:
	`org.relique.jdbc.csv.CsvDriver`

Parameter	**Description**

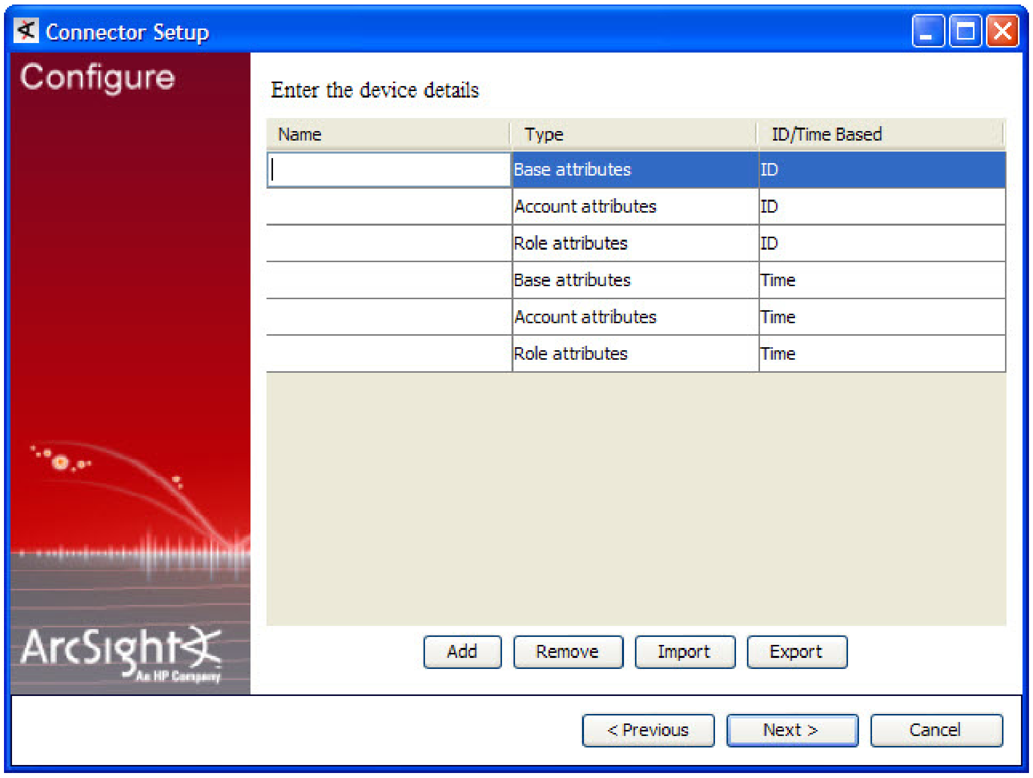
Database JDBC URL	Specify the URL with which to make a connection to your database using the driver specified in the Database Driver parameter.
	For Oracle, use:
	`jdbc:oracle:thin:@hostname_or_IP:1521:database_name`
	For MySQL, use:
	`jdbc:mysql://hostname_or_IP:3306/database_name`
	For ODBC, use:
	`jdbc:odbc:dsn_name`
	For Microsoft SQL Server, use:
	`jdbc:microsoft:sqlserver://host:port;databasename=name`
	For PostGreSQL, use:
	`jdbc:postgresql://host/database`
	For DB2 unified driver, use:
	`jdbc:db2:database_name`
	For DB2 Legacy CLI-based, use:
	`jdbc:db2://host_name:port_number/database_name`
	For Sybase, use:
	`jdbc:sybase:Tds:dbhost:dbport/dbname`
	For CSV, use:
	`jdbc:relique:csv:dir_of_csv_file`
	Use the absolute path to specify the location of the CSV input file.

Database Username	Enter the user name or user ID for the database.
Database Password	Enter the password for the user name or user ID.
Query Interval (in minutes)	Enter the time, in minutes, between queries to the database. The default is 1 minute.
IDM Identfier	String used to identify the IDM in the ESM console and group identities under one group named after the IDM Identifier. Once the IDM Identifier is set, you cannot change it without re-configuring the connector. If the IDM Identifier changes after initial import, ESM will not be able to relate ongoing updates to the right actor.

Note

The database username and password are null for the CSV driver. The CSV folder requires a metaData file with column types. See "CSV Requirements" on page 17 for more CSV specific requirements and file samples.

Enter the name, type and database type for the parsers required to retrieve identity information from the database tables.



Enter the device details

Name	Type	ID/Time Based
	Base attributes	ID
	Account attributes	ID
	Role attributes	ID
	Base attributes	Time
	Account attributes	Time
	Role attributes	Time

Property	Description
Name	Enter the name for the parser.
Type	Specifies the type of identity information being retrieved for the parser.
ID/Time Based	Specifies the type of database table the parser will query, either ID or time.

Tip

You can safely ignore any warning messages you receive stating that the database version could not be verified.

Note

The connector runs in the initial import mode when started for the first time. For subsequent restarts, it runs in update mode. Ensure that you add all the parsers you want before the intial start. If you want to add parsers after the initial start, set the internal parameter `initialimportstarted` to false. For more information, see "Reloading Actor Model Attributes" on page 23.

Parser templates created during the configuration process are located at:
`\user\agent\flexagent\mic\flexdatabase`

For more information about parser templates, see Chapter 4, Parser Templates, on page 25.

CSV Requirements

If you are using the connector to process CSV files, the following requirements must be met:

- Before installing the connector, download the CSV JDBC driver jar from http://csvjdbc.sourceforge.net/ and add it to the `user/agent/lib` directory. This release of the connector is certified with version 1.0.13 of the CSV JDBC jar.
- The CSV folder should have a file (named metaData) containing column type information. For example:

  ```
  columnTypes.actor=String,String,String,Timestamp
  columnTypes.roleCsv=String,String,String,Timestamp,String
  columnTypes.accountCsv=String,String,Timestamp,String
  ```

  Where actor, roleCsv and accountCsv are names of CSV files in the folder.
- In the CSV file, the first line must have the column names. For example:

  ```
  UniqueUserId,fullname,email,creationtime
  1234-1234-ABCD-CSV,Csvfullname1,name1@domain1.com,2012-04-15
  07:07:07
  1234-5678-ABCD-CSV,Csvfullname2,nam21@domain21.com,2012-04-15
  07:07:07
  1234-1234-XYZ-CSV,Csvfullname3,name3@domain3.com,2012-04-15
  07:07:07
  2134-5678-XYZ-CSV,Csvfullname4,name4@domain4.com,2012-04-15
  07:07:07
  1234-ABCD-CSV,Csvfullname5,name5@domain5.com,2012-04-15
  07:07:07
  ```

- For a time-based table follower, the time field must be of type Timestamp. For an ID-based table follower, the ID field must be of type Int.
- The CSV driver converts column names to upper case in the column meta data information in the result set. Since velocity is case sensitive, convert the fields to upper case. For example, even if your column name in the CSV file says creationTime, use `timestamp.field=CREATIONTIME`.
- When adding additional records, make sure the timestamp field or the ID field for a new record is later than the most recent record so the connector can detect and add new records.

Changing the Heap Size

If you are going to import a large number of actors, it is recommended that you increase the heap size of the connector. The default heap size is 256 MB. If you are going to run the connector as a service, set the heap size in the following file:
`../current/config/agent/agent.wrapper.conf`

Set the following parameters:

```
#Initial Java Heap Size (in MB)
wrapper.java.initmemory=2048
```

```
#Maximum Java Heap Size (in MB)
wrapper.java.maxmemory=5120
```

If the connector runs in standalone mode, the default heap size is 256 MB. For proper operation of the connector with a large number of actors, HP recommends that you modify the heap size setting to 4 GB. For 500,000 actors, the heap size should be 4 to 6 GB. Increase the memory for the connector by creating one of the following commands:

- For Linux - create the following shell script:
  `~ARCSIGHT_HOME/current/user/agent/setmem.sh`
  with the following content:
  `ARCSIGHT_MEMORY_OPTIONS=" -Xms4096m -Xmx6144m "`
- For Windows - create the following batch file:
  `$ARCSIGHT_HOME\current\user\agent\setmem.bat`
  with the following content:
  `SET ARCSIGHT_MEM_OPTIONS= -Xms4096m -Xmx6144m`

Note

- `ARCSIGHT_HOME` represents the directory where the connector is installed.
- Use regular double quote characters in the commands.

Tip

When working with large numbers of actors, accounts and rules, if you notice any discrepancies during import, check for database-related errors in the log. Environment issues such as slow database processing or high network traffic might be the cause. In that case, you might want to export the data to a CSV file and use CSV to import the data.
If the accounts and roles for an actor do not display in the console, check the EndTime attribute. If a value is set, accounts and roles are not displayed in the console. If you want the accounts and roles for an actor to display in the console, set the value to NULL.

Optional Optimization of Data Transfer

For medium to large deployments, it is recommended that the `sleepbetweenparsers` parameter be increased. For 500,000 actors, good performance was obtained by setting the value to 7,800,000 ms:

```
agents[0].sleepbetweenparsers=7800000
```

For smaller deployments (2500 actors), 60,000 ms provided good performance. The default value is 1000 ms. The `sleepbetweenparsers` parameter is located in the agent properties file.

Running Model Import Connectors

Model Import Connectors (MIC) can be installed and run in standalone mode, on Windows platforms as a Windows service, or on UNIX platforms as a UNIX daemon, depending upon the platform supported. On Windows platforms, connectors also can be run using shortcuts and optional Start menu entries.

If installed standalone, the connector must be started manually, and is not automatically active when a host is re-started. If installed as a service or daemon, the connector runs automatically when the host is re-started. For information about connectors running as services or daemons, see the ArcSight SmartConnector User's Guide.

For connectors installed standalone, to run all installed SmartConnectors and MIC connectors on a particular host, open a command window, go to `$ARCSIGHT_HOME\current\bin` and run: `arcsight connectors`

To view the SmartConnector log, read the file: `$ARCSIGHT_HOME\current\logs\agent.log`

To stop all SmartConnectors, enter `Ctrl+C` in the command window.

Set the Model Import User

After installing, configuring, and starting the connector, from the ArcSight ESM Console set the Model Import User for the connector (this can be admin or some other user).

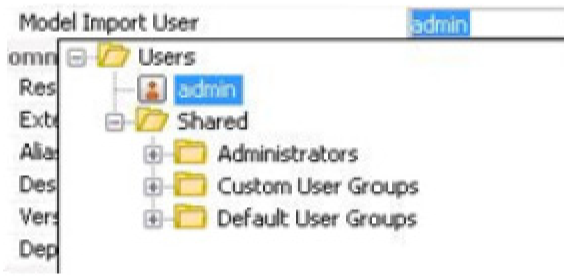
1 From the **ESM Console**, go to the **Navigator** panel and choose the **Resources** tab.

2 Under **Resources**, choose the **Connector** tab.

3 From under the **All Connector** directory, navigate to your Actor Model Import FlexConnector for Database.

4 Move to the **Inspect/Edit** panel and choose the **Connector** tab.

5 Under the **Connector** tab, go to **Model Import User** and select an admin user from the drop down list, as shown below:



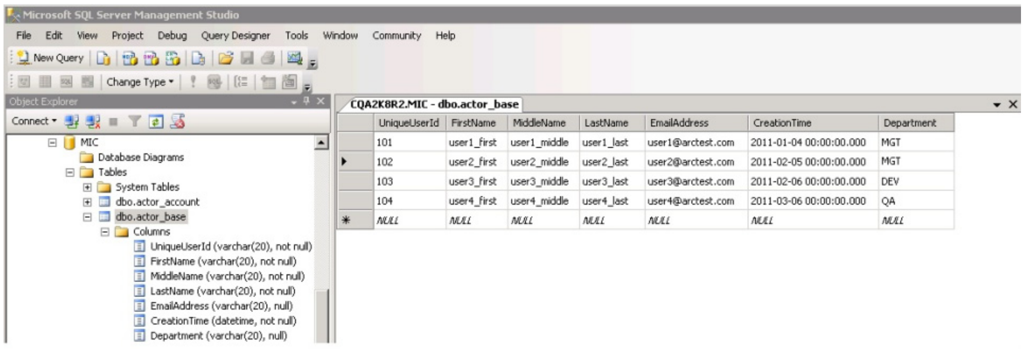
6 Click **OK**.

Parser Examples

This section provides examples of base, account, and role parsers.

Time-based Base Attributes Parser Example

The following is an example of a time-based base attributes parser and the table for which the query was written. For the initial import, it finds the changes from the beginning and remembers the timestamp of the last record it processes, which it uses to find changes after that time.


Microsoft SQL Server Management Studio
File Edit View Project Debug Query Designer Tools Window Community Help
New Query
Change Type
Object Explorer
Connect
CQA2K8R2.MIC - dbo.actor_base

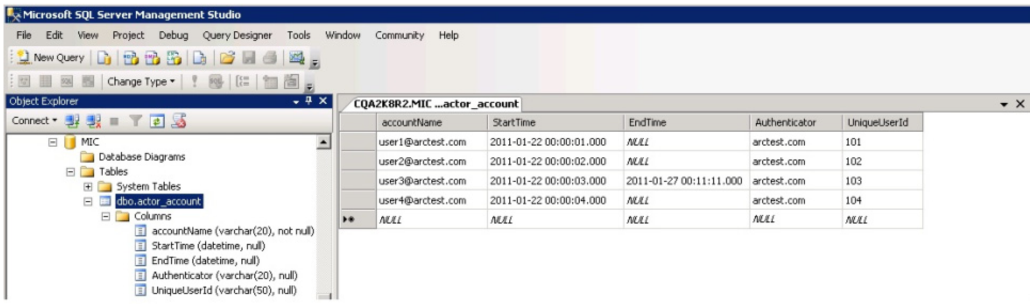
```
MIC
  Database Diagrams
  Tables
    System Tables
    dbo.actor_account
    dbo.actor_base
      Columns
        UniqueUserId (varchar(20), not null)
        FirstName (varchar(20), not null)
        MiddleName (varchar(20), not null)
        LastName (varchar(20), not null)
        EmailAddress (varchar(20), null)
        CreationTime (datetime, not null)
        Department (varchar(20), null)
```

	UniqueUserId	FirstName	MiddleName	LastName	EmailAddress	CreationTime	Department
	101	user1_first	user1_middle	user1_last	user1@arctest.com	2011-01-04 00:00:00.000	MGT
▶	102	user2_first	user2_middle	user2_last	user2@arctest.com	2011-02-05 00:00:00.000	MGT
	103	user3_first	user3_middle	user3_last	user3@arctest.com	2011-02-06 00:00:00.000	DEV
	104	user4_first	user4_middle	user4_last	user4@arctest.com	2011-03-06 00:00:00.000	QA
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
version.order=1

version.id=1

version.query=SELECT 1 as version from MIC.dbo.actor_base

lastdate.query=SELECT MAX(CreationTime) from MIC.dbo.actor_base

query=SELECT UniqueUserId, FirstName, MiddleName, LastName,
EmailAddress, CreationTime, Department \

FROM MIC.dbo.actor_base Where CreationTime > ?

timestamp.field=CreationTime

uniqueid.fields=CreationTime,UniqueUserId

###keep these 7 fields unchanged###

additionaldata.enabled=true

additionaldata.duplicate.keys.allowed=false

event.deviceEventCategory=__stringConstant("Actor")

event.deviceCustomString1Label=__stringConstant(model.sender)

event.deviceCustomString1=__stringConstant(flexdatabase)

event.deviceCustomString2Label=__stringConstant(model.template)

event.deviceCustomString2=__stringConstant(../flexagent/mic/flexda
tabase/base.vm)

###field mappings###

event.deviceVendor=__getVendor("Sql Server 2008 Database")

event.deviceProduct=__stringConstant(Identity Manager)

event.destinationUserId=UniqueUserId

###optional mappings###

additionaldata.StartTime=CreationTime
```

Time-based Account Attributes Parser

The following is an example of an SQL query for a time-based account attributes parser and the table for which the query was written:


Microsoft SQL Server Management Studio
File Edit View Project Debug Query Designer Tools Window Community Help
New Query
Change Type
Object Explorer
Connect
CQA2K8R2.MIC ...actor_account

MIC
Database Diagrams
Tables
System Tables
dbo.actor_account
Columns
accountName (varchar(20), not null)
StartTime (datetime, null)
EndTime (datetime, null)
Authenticator (varchar(20), null)
UniqueUserId (varchar(50), null)

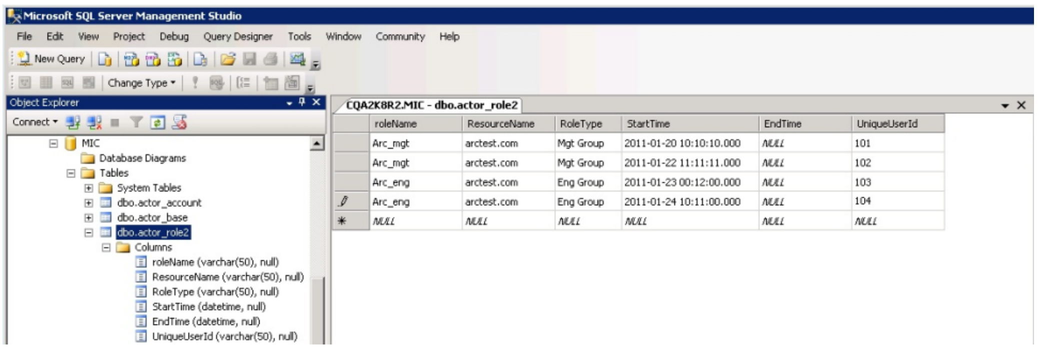
	accountName	StartTime	EndTime	Authenticator	UniqueUserId
	user1@arctest.com	2011-01-22 00:00:01.000	NULL	arctest.com	101
	user2@arctest.com	2011-01-22 00:00:02.000	NULL	arctest.com	102
	user3@arctest.com	2011-01-22 00:00:03.000	2011-01-27 00:11:11.000	arctest.com	103
	user4@arctest.com	2011-01-22 00:00:04.000	NULL	arctest.com	104
▸*	NULL	NULL	NULL	NULL	NULL

```
version.order=2

version.id=1

version.query=SELECT 1 as version from MIC.dbo.actor_account

lastdate.query=SELECT MAX(StartTime) from MIC.dbo.actor_account

query=SELECT accountName as Account, StartTime, EndTime,
Authenticator, UniqueUserId \

  FROM MIC.dbo.actor_account Where StartTime > ?

timestamp.field=StartTime

uniqueid.fields=StartTime,UniqueUserId

###keep these 7 fields unchanged###

additionaldata.enabled=true

additionaldata.duplicate.keys.allowed=false

event.deviceEventCategory=__stringConstant("Actor")

event.deviceCustomString1Label=__stringConstant(model.sender)

event.deviceCustomString1=__stringConstant(flexdatabase)

event.deviceCustomString2Label=__stringConstant(model.template)

event.deviceCustomString2=__stringConstant(../flexagent/mic/flexda
tabase/account.vm)

###field mappings###

event.deviceVendor=__getVendor("ktest SQL Server 2008 Database")

event.deviceProduct=__stringConstant(HP Identity Manager)

event.destinationUserId=UniqueUserId
```

```
###optional mappings###
```

Time-based Role Attributes Parser

The following is an example of an SQL query for a time-based role attributes parser and the table for which the query was written:


Microsoft SQL Server Management Studio
File Edit View Project Debug Query Designer Tools Window Community Help
New Query
Change Type
Object Explorer
Connect
CQA2K8R2.MIC - dbo.actor_role2

```
MIC
  Database Diagrams
  Tables
    System Tables
    dbo.actor_account
    dbo.actor_base
    dbo.actor_role2
      Columns
        roleName (varchar(50), null)
        ResourceName (varchar(50), null)
        RoleType (varchar(50), null)
        StartTime (datetime, null)
        EndTime (datetime, null)
        UniqueUserId (varchar(50), null)
```

roleName	ResourceName	RoleType	StartTime	EndTime	UniqueUserId
Arc_mgt	arctest.com	Mgt Group	2011-01-20 10:10:10.000	NULL	101
Arc_mgt	arctest.com	Mgt Group	2011-01-22 11:11:11.000	NULL	102
Arc_eng	arctest.com	Eng Group	2011-01-23 00:12:00.000	NULL	103
Arc_eng	arctest.com	Eng Group	2011-01-24 10:11:00.000	NULL	104
NULL	NULL	NULL	NULL	NULL	NULL

```
version.order=3

version.id=1

version.query=SELECT 1 as version from MIC.dbo.actor_role2

lastdate.query=SELECT MAX(StartTime) from MIC.dbo.actor_role2

query=SELECT roleName as Role, ResourceName, RoleType, StartTime,
EndTime, UniqueUserId \

  FROM MIC.dbo.actor_role2 Where StartTime > ?

timestamp.field=StartTime

uniqueid.fields=StartTime,UniqueUserId

###keep these 7 fields unchanged###

additionaldata.enabled=true

additionaldata.duplicate.keys.allowed=false

event.deviceEventCategory=__stringConstant("Actor")

event.deviceCustomString1Label=__stringConstant(model.sender)

event.deviceCustomString1=__stringConstant(flexdatabase)

event.deviceCustomString2Label=__stringConstant(model.template)

event.deviceCustomString2=__stringConstant(../flexagent/mic/flexda
tabase/role.vm)

###field mappings###
```

```
event.deviceVendor=__getVendor("ktest SQL Server 2008 Database")

event.deviceProduct=__stringConstant(HP Identity Manager)

event.destinationUserId=UniqueUserId

###optional mappings###
```

Reloading Actor Model Attributes

A redeployment, reconfiguration or mistaken deletion of attributes of your ESM structure may require reloading all actor model attributes. Use the following procedure to reload actor model attributes:

1 Stop the connector if running.

2 From the ESM Console, go to the **Navigator** panel and choose the **Resources** tab.

3 Under **Resources**, choose the **Actors** tab.

4 Under **All Actors**, go to the top level directory. The folder name should mimic the authenticator name of your actor data. Highlight the **actor data**, right-click and choose **Delete Group** from the shortcut menu. Do not delete actors outside of this top level directory.

Note

If you are deleting a large number of actors, see Open Issues for Systems with a Large Number of Actors (SOL-3525) in the IdentityView 2.52 Release Notes.

5 Delete all the files under the following directory: `$ARCSIGHT_HOME/user/agent/agentdata`

6 From the `$ARCSIGHT_HOME\current\bin` directory in a DOS command window, enter the following: `arcsight connectorsetup`

7 When the information message displays asking whether you want to enter Wizard mode, click **No**. The Agent Configuration Tool window displays.

8 From the **Options** menu, select **Show Internal Parameters**.

9 Change the **initialimportstarted** parameter to **false**.

10 Click **OK**.

11 Restart the connector.

Chapter 4

Parser Templates

This chapter provides information about the Actor Model Import FlexConnector for Database parser templates.

The following topics are covered:

Types of Parser Templates

The Actor Model Import FlexConnector for Database includes parser templates that are generated during connector configuration. There are three types of parser templates:

- **Base attributes** - use this template to load the base attributes into ESM. After creation, you edit the parser to add additional mappings for attributes. Parser location: `user/agent/flexagent/mic/flexdatabase/<base-name-provided>.sdktbdatabase.properties or sdkibdatabase.properties`
- **Account attributes** - use this template to load the authenticators and the accounts being authenticated. Both authenticator and account are required. After creation, you edit the parser to add additional mappings for attributes. Parser location: `user/agent/flexagent/mic/flexdatabase/<account-name-provided>/<account-name-provided>.sdktbdatabase.properties` for timebased or `sdkibdatabase.properties` for id based.
- **Role attributes** - use this template to load role names, resource names and role types. Only the role name is required. After creation, you edit the parser to add additional mappings for attributes. Parser location: `user/agent/flexagent/mic/flexdatabase/<role-name-provided>/<role-name-provided>.sdktbdatabase.properties` or `sdkibdatabase.properties`

Base Attributes Parsers

Base attributes parsers should provide the attributes in the following table in additional data.

Attribute	Description
FullName	**Required** The actor's full name as concatenated in the IDM or database.
FirstName	Specifies the actor's first name.
LastName	Specifies the actor's last name.
MiddleInitial	Specifies the actor's middle initial.
StartTime	If a time is not provided, it will default to system time.
DN	The distinguished name for the user, for example, CN=John Doe, OU=Sales, DC=companyname,DC=com
EmployeeType	The type of employee this actor is in your company. This value is usually a classification unique to your company's personnel operations, for example, full-time, exempt, or contractor.
Status	The employment status of the actor, one of: Active, Deleted or Disabled. When an actor is deleted from the IDM or database, the actor will remain in the ESM actor model with the status of deleted. This will preserve any history related to this actor in case activity appears on the system that is inappropriate to the actor's status. If the actor is deleted directly from ESM, the actor will be completely removed from the ESM actor model without preserving a history.
Title	Specifies the actor's job title.
Company	The company by whom the actor is employed, applies to contractors or employees from partner companies.
Org	The organization within your company of which the actor is a member.
Department	The actor's department.
Manager	The actor's manager.
Assistant	The actor's assistant.
EmailAddress	The actor's company email address.
Location	The actor's work location.
Office	The actor's office address.
BusinessPhone	The actor's business phone.
MobilePhone	The actor's mobile phone.
Fax	The actor's fax number.
Pager	The actor's pager number.
Address	Actor's business street address.

Attribute	Description
City	Actor’s business address city.
State	Actor’s business address state.
ZIPCode	Actor’s business address ZIP code.
CountryOrRegion	Actor’s business address country or region.

Note

The IDMIdentifier is captured during the connector setup. The UUID is mapped from event.destinationUserId.

Note that the UUID is not the same as uniqueid.fields. uniqueid.fields identifies a unique row in the database.

Base Attributes Parser ID-based Template

The following is the base attributes template for an ID database table:

```
version.order=1

version.id=1

version.query=<query to verify database version or existence of
certain table/columns>

maxid.query=<query to select maxid>

query=<select query to select appropriate columns. The query would
select all the records where id is greater than a given id--as a
parameter>

id.field=<id column>

###optional###

#uniqueid.fields=<comma separated field(s) identifying unique row
when id is same for than one row>

###keep these 7 fields unchanged###

additionaldata.enabled=true

additionaldata.duplicate.keys.allowed=false

event.deviceEventCategory=__stringConstant("Actor")

event.deviceCustomString1Label=__stringConstant(model.sender)

event.deviceCustomString1=__stringConstant(flexdatabase)

event.deviceCustomString2Label=__stringConstant(model.template)

event.deviceCustomString2=__stringConstant(../flexagent/mic/flexda
tabase/base.vm)

###field mappings###

event.deviceVendor=__getVendor("My Database")

event.deviceProduct=__stringConstant(Identity Manager)
event.destinationUserId=<user id column>

###optional mappings###
```

Note: Edit the version.query, lastdate.query and query to reflect your database tables. Enter the timestamp and uniqueid fields and the event.destinationUserId.

Base Attributes Parser Time-based Template

The following is the base attributes parser template for timestamp database tables:

```
version.order=1

version.id=1

version.query=<query to verify database version or existence of
certain table/columns>

lastdate.query=<query to select max timestamp>

query=<select query to select appropriate columns. The query would
select all the records where timestamp is greater than a given
timestamp--as a parameter>

timestamp.field=<timestamp column>

uniqueid.fields=<comma separated field(s) identifying unique row>

###keep these 7 fields unchanged###

additionaldata.enabled=true

additionaldata.duplicate.keys.allowed=false

event.deviceEventCategory=__stringConstant("Actor")

event.deviceCustomString1Label=__stringConstant(model.sender)

event.deviceCustomString1=__stringConstant(flexdatabase)

event.deviceCustomString2Label=__stringConstant(model.template)

event.deviceCustomString2=__stringConstant(../flexagent/mic/flexda
tabase/base.vm)

###field mappings###

event.deviceVendor=__getVendor("My Database")

event.deviceProduct=__stringConstant(Identity Manager)

event.destinationUserId=<user id column>

###optional mappings###
```

Account Parsers

This section provides information about the attributes that account parsers must provide the connector and account parser templates.

Account parsers must provide the following attributes in additional data:

Attribute	Desciption
StartTime	This attribute must be provided for both ID-based and Time-based parsers.
EndTime	If a time is not provided, it will be an open time. If you want to disassociate an actor from an account, the EndTime must be specified. If you want the accounts and roles for an actor to display in the console, set the value to NULL.
Authenticator	**Required** - This is the friendly name for the user authentication data store derived by the connector based on information in the IMS or database, for example: “Active Directory:mycompany.com”, “Oracle”.
Account	**Required** - This is attribute contains all the user’s account IDs tracked in the authentication data store, for example, john_doe, jdoe, or john.d. It is required.

Note

The UUID is mapped from event.destinationUserId.

Note that the UUID is not the same as uniqueid.fields. uniqueid.fields identifies a unique row in the database.

Account Attributes Parser ID-based Template

The following is the account attributes parser template for ID-based database tables:

```
version.order=2

version.id=1

version.query=<query to verify database version or existence of
certain table/columns>

maxid.query=<query to select maxid>

query=<select query to select appropriate columns. The query would
select all the records where id is greater than a given id--as a
parameter>

id.field=<id column>

###optional###

#uniqueid.fields=<comma separated field(s) identifying unique row
when id is same for than one row>

###keep these 7 fields unchanged###
```

```
additionaldata.enabled=true

additionaldata.duplicate.keys.allowed=false

event.deviceEventCategory=__stringConstant("Actor")

event.deviceCustomString1Label=__stringConstant(model.sender)

event.deviceCustomString1=__stringConstant(flexdatabase)

event.deviceCustomString2Label=__stringConstant(model.template)

event.deviceCustomString2=__stringConstant(../flexagent/mic/flexda
tabase/account.vm)

###field mappings###

event.deviceVendor=__getVendor("My Database")

event.deviceProduct=__stringConstant(Identity Manager)

event.destinationUserId=<user id column>

###optional mappings###
```

Account Attributes Parser Time-based Template

The following is the account attributes parser template for timestamp database tables:

```
version.order=2

version.id=1

version.query=<query to verify database version or existence of
certain table/columns>

lastdate.query=<query to select max timestamp>

query=<select query to select appropriate columns. The query would
select all the records where timestamp is greater than a given
timestamp--as a parameter>

timestamp.field=<timestamp column>

uniqueid.fields=<comma separated field(s) identifying unique row>

###keep these 7 fields unchanged###

additionaldata.enabled=true

additionaldata.duplicate.keys.allowed=false

event.deviceEventCategory=__stringConstant("Actor")

event.deviceCustomString1Label=__stringConstant(model.sender)

event.deviceCustomString1=__stringConstant(flexdatabase)

event.deviceCustomString2Label=__stringConstant(model.template)
event.deviceCustomString2=__stringConstant(../flexagent/mic/flexda
tabase/account.vm)

###field mappings###

event.deviceVendor=__getVendor("My Database")

event.deviceProduct=__stringConstant(Identity Manager)

event.destinationUserId=<user id column>

###optional mappings###
```

Role Parsers

This section provides information about the attributes that role parsers must provide the connector and role parser templates.

Role parsers must provide the following attributes in additional data:

Attribute	Description
StartTime	This attribute must be provided for both ID-based and Time-based parsers.
EndTime	If a time is not provided, it will be an open time. If you want to disassociate an actor from a role, the EndTime must be specified. If you want the accounts and roles for an actor to display in the console, set the value to NULL.
Role	**Required** - Name of the role, such as Manager or Administrator.
ResourceName	Name of the application, organization, or network resource for which that person performs the role, such as the name of your company, the name of the application to which the user has privileges, or the name of a network device to which the user has privileges.
RoleType	The role type is the role’s category. For example, Global Security Group or Local Distribution Group.

Note

The UUID is mapped from event.destinationUserId.

Note that the UUID is not the same as uniqueid.fields. uniqueid.fields identifies a unique row in the database.

Role Attributes Parser ID-based Template

The following is the role attributes parser template for ID-based database tables:

```
version.order=3

version.id=1

version.query=<query to verify database version or existence of
certain table/columns>
```

```
maxid.query=<query to select maxid>

query=<select query to select appropriate columns. The query would
select all the records where id is greater than a given id--as a
parameter>

id.field=<id column>

###optional###

#uniqueid.fields=<comma separated field(s) identifying unique row
when id is same for than one row>

###keep these 7 fields unchanged###

additionaldata.enabled=true

additionaldata.duplicate.keys.allowed=false

event.deviceEventCategory=__stringConstant("Actor")

event.deviceCustomString1Label=__stringConstant(model.sender)

event.deviceCustomString1=__stringConstant(flexdatabase)

event.deviceCustomString2Label=__stringConstant(model.template)

event.deviceCustomString2=__stringConstant(../flexagent/mic/flexda
tabase/role.vm)

###field mappings###

event.deviceVendor=__getVendor("My Database")

event.deviceProduct=__stringConstant(Identity Manager)

event.destinationUserId=<user id column>

###optional mappings###
```

Role Attributes Parser Time-based Template

The following is the role attributes parser template for timestamp database tables:

```
version.order=3

version.id=1

version.query=<query to verify database version or existence of
certain table/columns>

lastdate.query=<query to select max timestamp>

query=<select query to select appropriate columns. The query would
select all the records where timestamp is greater than a given
timestamp--as a parameter>

timestamp.field=<timestamp column>

uniqueid.fields=<comma separated field(s) identifying unique row>

###keep these 7 fields unchanged###
```

```
additionaldata.enabled=true

additionaldata.duplicate.keys.allowed=false

event.deviceEventCategory=__stringConstant("Actor")

event.deviceCustomString1Label=__stringConstant(model.sender)

event.deviceCustomString1=__stringConstant(flexdatabase)

event.deviceCustomString2Label=__stringConstant(model.template)

event.deviceCustomString2=__stringConstant(../flexagent/mic/flexda
tabase/role.vm)

###field mappings###

event.deviceVendor=__getVendor("My Database")

event.deviceProduct=__stringConstant(Identity Manager)

event.destinationUserId=<user id column>

###optional mappings###
```