

# **Tivoli Identity Manager 5.1**

## **Data synchronization Execution Sequence**

Data synchronization consists of the following steps:

### **1. Scheduling Data Synchronization**

When the system administrator creates Data Synchronization schedule, an entry is made in the RESOURCES\_SYNCHRONIZATIONS table to save the time of the schedule. A message is also sent to the Scheduler Bean, which contains the recurring schedule object and the destination JMS queue.

### **2. Execute Data Synchronization**

- A. Once we schedule a full data synchronization operation, an entry gets created in SCHEDULED\_MESSAGE table. Every node in the ITIM cluster runs a daemon or server thread called scheduler that periodically scans the SCHEDULED\_MESSAGE table and schedules the messages that are due. Only one node of the cluster will process an entry so the data synchronization operation will get scheduled on one node of the cluster. Once scheduled, the scheduler will send a message to JMS queue for processing. `AdhocSyncDataMessageReceiver.onMessage()` is main JMS handler method which receives the message from the JMS queue (adhocSyncQueue) and starts its execution. It receives synchronization unit as a parameter and starts data synchronization of selected sync unit.
- B. To start the Data Synchronization the TIM Server has to acquire the lock. Before acquiring the lock the TIM Server tries to write its hostname to this table and the table is locked. This prevents other TIM Servers from writing to this table. In `AdhocSynchronizationManagerLocal.acquireLock()` method, first lock entry is added in SYNCHRONIZATION\_LOCK table. (This table is used to avoid race condition when two TIM Servers in a clustered environment start Data Synchronizations at the same time. ) After that in `createHistoryEntry()` method an entry in the history table of synchronization. i.e. SYNCHRONIZATION\_HISTORY table. It set synchronization status as 'started'. Following debug statement is logged at this step:-

**(Converted DEBUG\_MAX to DEBUG\_MID statement)**

```
<Trace Level="MID">
<Time Millis="1296767215000"> 2011.02.04 02:36:55.000+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport.synchronization</Component>
```

```

<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[INSERT INTO itimuser.SYNCHRONIZATION_HISTORY (SYNC_ID, STATUS,
    STARTED_TIME, COMPLETED_TIME, SCHEDULED_TIME, REQUESTOR, REQ_TYPE, REQ_NAME,
    TENANT, STATUS_DETAIL, SERVER_NAME) VALUES (0, 'Started', 1296767215000, 1296767215000, 0,
    'ITIM Manager', 'DS', 'Data Synchronization', 'ou=ITIM51,dc=com', ?,
    'itimwin3Node02Cell/itimwin3Node02/server1') - with status_detail = ]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.synchronization.AdhocSynchronizationManagerBean"
    Method="createHistoryEntry"/>
<Thread>Default : 1</Thread>
</Trace>

```

AdhocSynchronizationManagerBean.synchronizeData() is a method which starts actual synchronization process. This method is also responsible for updating entries in the history table in the DB. This method logs the following debug statement as “Data synchronization called” **(Added New DEBUG\_MIN statement)**

```

<Trace Level="MIN">
<Time Millis="1297955672046"> 2011.02.17 20:44:32.046+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport.synchronization</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[Data synchronization called ...]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.synchronization.AdhocSynchronizationManagerBean"
    Method="synchronizeData"/>
<Thread>Default : 1</Thread>
</Trace>

```

- C. ‘Separation of Duty Policy’ behavior is introduced in ITIM 5.1 version. In data synchronization, new methods are added to synchronize SoD policies, rules and violations.

SynchronizePolicies.run(DistinguishedName) method executes to synchronize SoD policy by evaluating all rules against all those role members.

- D. The actual data synchronization process consists of dropping the existing Entity and ACI tables and recreating and populating them. The ENTITY\_COLUMN table contains the information needed to drop, create and populate these tables. The AVAILABLE\_FOR\_REPORTING field of the ENTITY\_COLUMN table contains the state of every attribute that was/is mapped. The different states in which an attribute can exist are as follows:

- NEW: A newly mapped attribute.
- DELETED\_NEW: An unmapped NEW attribute.
- AVAILABLE: An attribute available for reporting subject to success of the previous synchronization.

- DELETED\_ AVAILABLE: An unmapped attribute previously in an AVAILABLE state.
- TABLE\_DROPPED: An intermediate state for synchronization. Only a mapped attribute can attain this state.
- TABLE\_CREATED: An intermediate state for synchronization. Only a mapped attribute can attain this state.

AdhocSynchronizationManagerBean.stageData(DistinguishedName, Connection) method stages the ENTITY data to the database. This method will populate the ENTITY tables in the database with data from the ITIM directory server. ITIM APIs will be used to fetch the ITIM directory server data. In this method it reads 'commitFrequency' property from adhocReporting.properties file. It Checks if the value set for 'commitFrequency' > 1 then the number of uncommitted database updates 'numberOfUpdates' are incremented and commit is done everytime when 'numberOfUpdates' is equal to or more than 'commitFrequency'. In this method all database entities like views, tables gets deleted, then database schema is created and tables are populated with data from directory server.

To drop reporting schema, below sequence is followed

Helper.dropReportingViews() – This method drops all reporting views. When all reporting views gets deleted, following debug statement is logged: **(Added New DEBUG\_MID statement)**

```
<Trace Level="MID">
<Time Millis="1297989137640"> 2011.02.18 06:02:17.640+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.adhocreport.util</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[Dropped reporting views]]></LogText>
<Source FileName="com.ibm.itim.adhocreport.util.Helper" Method="dropReportingViews"/>
<Thread>Default : 2</Thread>
</Trace>
```

a) AdhocSynchronizationManagerBean.deleteStagedData(Connection) – This method deletes staged ENTITY data using following methods

a. AdhocSynchronizationManagerBean.deleteDeletedNewTables(Connection) : 'DELETED\_NEW' entries of ENTITY\_COLUMN table are unmapped new attributes. No processing needs to be done on this attribute during data synchronization. So in this method 'deleted\_new' table entries getting deleted from ENTITY\_COLUMN table. Following log statement is logged at this stage :

**(Converted DEBUG\_MAX to DEBUG\_MID statement)**

<Trace Level="MID">

<Time Millis="1297989291750"> 2011.02.18 06:04:51.750+05:30</Time>

<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>

<ProductId>CTGIM</ProductId>

<Component>com.ibm.itim.apps.ejb.adhocreport.synchronization</Component>

<ProductInstance>server1</ProductInstance>

<LogText><![CDATA[DELETE FROM itimuser.ENTITY\_COLUMN WHERE AVAILABLE\_FOR\_REPORTING  
= 'deleted\_new']]></LogText>

<Source FileName="com.ibm.itim.apps.ejb.adhocreport.synchronization.AdhocSynchronizationManagerBean"  
Method="deleteDeletedNewTables"/>

<Thread>Default : 2</Thread>

</Trace>

- b. AdhocSynchronizationManagerBean.deleteDeletedAvailTables(Connection) : DELETED\_AVAILABLE attribute is an unmapped attribute previously in an AVAILABLE state. During data synchronization, these attribute entries getting deleted from ENTITY\_COLUMN table and ENITITY tables created for these attributes also gets deleted. deleteDeletedAvailTables() method deletes entity tables having status 'deleted\_available' . Firstly following query is executed to retrieves all the entities which shows status as 'deleted\_available' from ENTITY\_COLUMN table:

<Trace Level="MAX">

<Time Millis="1297989327406"> 2011.02.18 06:05:27.406+05:30</Time>

<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>

<ProductId>CTGIM</ProductId>

<Component>com.ibm.itim.apps.ejb.adhocreport.synchronization</Component>

<ProductInstance>server1</ProductInstance>

<LogText><![CDATA[select distinct ENTITY\_NAME from itimuser.ENTITY\_COLUMN where  
(AVAILABLE\_FOR\_REPORTING = 'deleted\_available')]]></LogText>

<Source FileName="com.ibm.itim.apps.ejb.adhocreport.synchronization.AdhocSynchronizationManagerBean"  
Method="getColumns"/>

<Thread>Default : 2</Thread>

</Trace>

- c. AdhocSynchronizationManagerBean.deleteTableCreatedTables(Connection) TABLE\_CREATED status of entity means that the table has been created but the data is yet to be populated. This method first finds out, which tables need to be deleted. Following query is executed to get entity names from ENTITY\_COLUMN table which status as 'table\_created':

```
<Trace Level="MAX">
<Time Millis="1297989605687"> 2011.02.18 06:10:05.687+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport.synchronization</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[select distinct ENTITY_NAME from itimuser.ENTITY_COLUMN where
(AVAILABLE_FOR_REPORTING = 'table_created')]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.synchronization.AdhocSynchronizationManagerBean"
Method="getColumns"/>
<Thread>Default : 2</Thread>
</Trace>
```

This method drops all child tables created for multivalued attributes and then parent table whose status is 'table\_created'. It will also delete ACI related tables associated with these entities. Finally ENTITY\_COLUMN table is updated to set status of these entities as 'table\_dropped'.

- d. AdhocSynchronizationManagerBean.deleteAvailableTables(Connection): 'AVAILABLE' status shows that entity is available for reporting subject to success of the previous synchronization. deleteAvailableTables() method first find out table entities from ENTITY\_COLUMN table which are 'AVAILABLE' for reporting with following query :

```
<Trace Level="MAX">
<Time Millis="1297991389625"> 2011.02.18 06:39:49.625+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport.synchronization</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[select distinct ENTITY_NAME from itimuser.ENTITY_COLUMN where
(AVAILABLE_FOR_REPORTING = 'available')]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.synchronization.AdhocSynchronizationManagerBean"
Method="getColumns"/>
<Thread>Default : 2</Thread>
</Trace>
```

This method drops all child tables created for multivalued attributes for an entity. It will also delete

ACI related tables associated with each entity and finally parent entity table gets deleted table whose status is 'available'. ENTITY\_COLUMN table is updated to set status of these entities as 'table\_dropped'.

- E. The next step is to populate all tables. ACI related tables are getting populated first. The ACIs will be stored in the form of relationship tables between the user and the container entries. It store information about the specific access rights in the database. ACISynchronization.populateParsedACIs() method is invoked to populate ACI tables. Following log statement gets logged when population of ACI table starts: **(Added New DEBUG\_MIN statement)**

```
<Trace Level="MIN">
  <Time Millis="1297992081359"> 2011.02.18 06:51:21.359+05:30</Time>
  <Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
  <ProductId>CTGIM</ProductId>
    <Component>com.ibm.itim.apps.ejb.adhocreport</Component>
  <ProductInstance>server1</ProductInstance>
    <LogText><![CDATA[Populating parsed ACI related tables...]]></LogText>
      <Source FileName="com.ibm.itim.apps.ejb.adhocreport.ACISynchronization"
Method="populateParsedACIs"/>
    <Thread>Default : 2</Thread>
  </Trace>
```

First it flushes out entries from ACI table and its related tables in

ACISynchronization.deleteParsedACIEntries(Connection) method. Here data will be deleted from 'ACI', 'ACI\_ROLEDNS', 'ACI\_PRINCIPALS', 'ACI\_PERMISSION\_ATTRIBUTERIGHT' and 'ACI\_PERMISSION\_CLASSRIGHT' tables. In ACISynchronization.stageParsedACIs() method, it populates the ACI related tables. This method checks if, for an accessright object the filter is NULL or not. If the filter is not NULL then,

- 1) It gets the list of attributes for this filter. LDAP to SQL filter parser will do the job. Also, the targetClass for this AccessRight object should be used to map it to some entityName.
- 2) If this entity and the attributes are not present in the ENTITY\_COLUMN table, then they get added in ENTITY\_COLUMN table.

ACISynchronization.unmapImplicitAttributesNotRequired() method marks all the implicit attributes as "U" in the ENTITY\_COLUMN table which were implicitly mapped previously but no longer being used by any ACI in the system.

- F. Next step is to populate authorization owners. ACISynchronization.populateAuthorizationOwners() method populate AUTHORIZATION\_OWNERS table. This information will be used while generating ACI Report. When population of AUTHORIZATION\_OWNERS table starts, it logs following debug

statement : **(Added New DEBUG\_MID statement)**

```
<Trace Level="MID">
<Time Millis="1298015608156"> 2011.02.18 13:23:28.156+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[Populating AUTHORIZATION_OWNER table...]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.ACISynchronization"
Method="populateAuthorizationOwners"/>
<Thread>Default : 0</Thread>
</Trace>
```

Here it first delete entries from AUTHORIZATION\_OWNERS table, then invokes

createcachesForAllContainers() methods which create the cache having details about containers.

In supervisorCache HashMap Location/OU container DN is the key and collection of

supervisors(SystemUserEntity) are the values. In adminDomainCache

AdminDomain/OU/Location/BPO/Org container DN are the key and collection of Domain

Administrators(SystemUserEntity) are the values. In containerCache HashMap Parent container

DN is the key having value as a collection of children container. After that it gets a list of system

users who can do reporting. This is the set of system users, who are covered by reporting ACIs

defined in the system. ProcessAORRecursive() method is called recursively to populate

AUTHORIZATION\_OWNERS table for each system user who can do reporting. Following log

statement is logged when population of authorization owner table completes: **(Converted**

**DEBUG\_MAX to DEBUG\_MID statement)**

```
<Trace Level="MID">
<Time Millis="1298016806687"> 2011.02.18 13:43:26.687+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport.synchronization</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[AUTHORIZATION_OWNER Table Population Took : 1406328
milliseconds]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.synchronization.AdhocSynchronizationManagerBean"
Method="stageData"/>
<Thread>Default : 0</Thread>
</Trace>
```

G. If any ACI filter attributes or relationship attributes are missed, these attributes are added in

ENTITY\_COLUMN table in createFilterACIEntitythatColumnEntry() method.

The next step of data synchronization is to create staged tables.

AdhocSynchronizationManagerBean.createStagedTables(Connection) method finds out the tables to be created and creates them. This method updates ENTITY\_COLUMN table to set status for newly created table as 'created\_new'. And then find out what all tables need to be created in getColumnsForTableCreation() method. This method also creates ACI related tables like classright and attributeright tables for ACI. While creating tables for data synchronization, some reserve keywords cannot be used. The code reads the list of reserve keywords from the property "reservedWords" from adhocreporting.properties file.

Also if the entity/attribute name contains one or more of the disallowed characters specified in "disallowedChars" in adhocreporting.proeptriesm then these characters will be removed from the table/column name.

The characters specified in "disallowedCharsForStart" in adhocreporting.properties will not be used as the starting character of table/column name.

H. If change log is enabled in adhocReporting.properties method, then Helper.setChangeNumber() method, sets the changeNumber of the last processed changelog entry in the database (CHANGELOG table).

I. **Populating database tables** depending on ENTITY\_COLUMN table status.

AdhocSynchronizationManagerBean.populateDatabase() method is invoked to populate database.

i. If the value of 'availableForNonAdministrators' property is set to true, then ACI permissions related data is synchronized. Property value, 'availableForNonAdministrators = false' indicate that reporting data will be exclusively used by administrators and hence processing/population of user ACI permissions is unnecessary.

To synchronize ACIs ( if 'availableForNonAdministrators = true' ), it first finds out all target classes from ENTITY\_COLUMN table (according to which entities are available in ENTITY\_COLUMN table). It then creates the accessCache for a container having accessRightsCollection. The accessCache Format is :-

[(Key) targetClass]

[(Value) HashMap whose format is as follows:-

[(Key) self/supervisor/administrator/roleDN]

[(Value) Collection of AccessRights having the "key" as one of its principals.]

It logs following statement in trace.log:



**(Added New DEBUG\_MIN statement)**

```
<Trace Level="MIN">
<Time Millis="1298246062687"> 2011.02.21 05:24:22.687+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport.synchronization</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[Synchronizing ACIs ... generating dirAccessCache]]></LogText>
<Source
    FileName="com.ibm.itim.apps.ejb.adhocreport.synchronization.AdhocSynchronizationManagerBean"
    Method="populateDatabase"/>
<Thread>Default : 1</Thread>
</Trace>
```

Next step is to Populate parsed Entitlements – This debug statement is logged which shows synchronization current status - **(Added New DEBUG\_MIN statement)**

```
<Trace Level="MIN">
<Time Millis="1298246789890"> 2011.02.21 05:36:29.890+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport.synchronization</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[Synchronizing Entitlements ...]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.synchronization.AdhocSynchronizationManagerBean"
Method="populateDatabase"/>
<Thread>Default : 1</Thread>
</Trace>
```

SynchronizationHelper.populateParsedEntitlements(conn) method populates the parsed entitlement related tables. This method first deletes data from ENTITLEMENT and ENTITLEMENT\_PROVISIONINGPARAMS tables. After that it populates provisioning policy entitlements in ENTITLEMENT table and provisioning parameters in ENTITLEMENT\_PROVISIONINGPARAMS table. Following Insert SQL statement of ENTITLEMENT table is logged in trace.log :

```
<Trace Level="MAX">
<Time Millis="1298247581890"> 2011.02.21 05:49:41.890+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
```

```

<Component>com.ibm.itim.apps.ejb.adhocreport</Component>

<ProductInstance>server1</ProductInstance>

<LogText><![CDATA[insert into itimuser.Entitlement (dn, type, servicetargettype, servicetargetname,
processDN) values (?, ?, ?, ?, ?)]]></LogText>

<Source FileName="com.ibm.itim.apps.ejb.adhocreport.SynchronizationHelper"
Method="populateParsedEntitlements"/>

<Thread>Default : 1</Thread>

</Trace>

```

Following Insert statement of ENTITLEMENT\_PROVISIONINGPARAMS table is logged in trace.log:

```

<Trace Level="MAX">

<Time Millis="1298247871828"> 2011.02.21 05:54:31.828+05:30</Time>

<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>

<ProductId>CTGIM</ProductId>

<Component>com.ibm.itim.apps.ejb.adhocreport</Component>

<ProductInstance>server1</ProductInstance>

<LogText><![CDATA[insert into itimuser.Entitlement_ProvisioningParams (dn, name, attributevalue, enforcement,
exprType) values (?, ?, ?, ?, ?)]]></LogText>

<Source FileName="com.ibm.itim.apps.ejb.adhocreport.SynchronizationHelper"
Method="populateParsedEntitlements"/>

<Thread>Default : 1</Thread>

</Trace>

```

When it completes with entitlement synchronization, it logs following debug statement

**(Added New DEBUG\_MID statement)**

```

<Trace Level="MID">

<Time Millis="1298248052093"> 2011.02.21 05:57:32.093+05:30</Time>

<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>

<ProductId>CTGIM</ProductId>

<Component>com.ibm.itim.apps.ejb.adhocreport</Component>

<ProductInstance>server1</ProductInstance>

<LogText><![CDATA[Populated parsed entitlements related tables.]]></LogText>

<Source FileName="com.ibm.itim.apps.ejb.adhocreport.SynchronizationHelper"
Method="populateParsedEntitlements"/>

<Thread>Default : 1</Thread>

</Trace>

```

ii.To populate all ENTITY tables, it first find out all containers in

[SynchronizationHelper.getAllContainers\(\)](#) method. This method will return a collection of objects in the directory. This Object type definition depends on the category of the entity or DNs of the Container entity object. Then [SynchronizationHelper.populateIndexAttributesMap\(\)](#) method creates attribute index hashmap which will be used during Entity population. It creates index hashmap for indexes mentioned in `adhocreporing.properties# reportIndexes` property.

iii. Now according to entity list which is retrieved from ENTITY\_COLUMN table, we need to populate each entity. [SynchronizationHelper.populateEntity\(\)](#) method populates each entity from ENTITY\_COLUMN. First, it sets entity status as 'available' in ENTITY\_COLUMN table - Following debug statement is logged which shows SQL query to update ENTITY\_COLUMN table :

**(Converted DEBUG\_MAX to DEBUG\_MID statement)**

```
<Trace Level="MID">
<Time Millis="1298248756984"> 2011.02.21 06:09:16.984+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[UPDATE itimuser.ENTITY_COLUMN SET AVAILABLE_FOR_REPORTING = 'available'
WHERE LOWER(ENTITY_NAME) = LOWER('ADFeed')]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.SynchronizationHelper" Method="populateEntity"/>
<Thread>Default : 1</Thread>
</Trace>
```

It finds out columns for each entity and separates column list in single valued and multivalued attributes. For any entity, only mapped attributes are returned. But for organizationalContainers all attributes are returned.

iv. Each entity is processed in [SynchronizationHelper.populateEntity\(\)](#) method . If selected entity is organization container, then it invokes [getOrganizationalContainerData\(\)](#) method which retrieve all container objects in given container i.e. organizational unit, location, Business partner organization and admin domain. All other entity data is fetched using [SynchronizationHelper.fetchDirectoryData\(\)](#) method. This method fetches the data from the ITIM directory server for the given ENTITY. It uses ITIM APIs for fetching the data from Directory Server. If the value of 'normalizedDN' property in `adhocreporing.properties` is set to 'true', all DN attributes will be normalized during data synchronization

When data retrieval is complete for each entity, it logs following log statements:

**(Converted DEBUG\_MAX to DEBUG\_MID statement)**

```
<Trace Level="MID">
```

```

<Time Millis="1298249274765"> 2011.02.21 06:17:54.765+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[Data Population for ADFeed in this container OVER]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.SynchronizationHelper" Method="processEntity"/>
<Thread>Default : 1</Thread>
</Trace>

```

```

<Trace Level="MID">
<Time Millis="1298249283921"> 2011.02.21 06:18:03.921+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[0 data populated in msecs: 4625]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.SynchronizationHelper" Method="processEntity"/>
<Thread>Default : 1</Thread>
</Trace>

```

v. According to entity columns, for single valued attribute parent table of each entity is populated. If there is any multivalued attribute present, child table for this entity is populated accordingly.

'sqlBatchSize' property from adhocreporting.properties, is used to insert number of records in one SQL Batch.

vi. If value for 'createIndex' property from adhocreporting.properties is set to 'true' then, indexes have been created for entity tables. Following debug statement is logged at this stage:

```

<Trace Level="MAX">
<Time Millis="1298250299015"> 2011.02.21 06:34:59.015+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[create index TIM828186159951847 on ADFeed (CONTAINERDN asc)]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.SynchronizationHelper" Method="createSingleIndices"/>
<Thread>Default : 1</Thread>
</Trace>

```

When entity population completes it logs following debug statement;

**(Converted DEBUG\_MAX to DEBUG\_MID statement)**

```

<Trace Level="MID">
<Time Millis="1298250694640"> 2011.02.21 06:41:34.640+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>

```

```

<Component>com.ibm.itim.apps.ejb.adhocreport.synchronization</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[Entity Table Population Took : 1966625 milliseconds]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.synchronization.AdhocSynchronizationManagerBean"
Method="populateDatabase"/>
<Thread>Default : 1</Thread>
</Trace>

```

- vii. SynchronizationHelper.populateServiceAccountInfo() method populates service profiles and their corresponding account profiles in a table called SERVICE\_ACCOUNT\_MAPPING.

Following debug statement is logged at this stage of synchronization:

**(Converted DEBUG\_MAX to DEBUG\_MID statement)**

```

<Trace Level="MID">
<Time Millis="1298250694640"> 2011.02.21 06:41:34.640+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport.synchronization</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[Populating SERVICE_ACCOUNT_MAPPING table ...]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.synchronization.AdhocSynchronizationManagerBean"
Method="populateDatabase"/>
<Thread>Default : 1</Thread>
</Trace>

```

- SynchronizationHelper.populateRecertifierDetails() method populates

RECERERTIFIER\_DETAILS\_INFO table. Following debug statement is logged at this stage of synchronization: **(Added New DEBUG\_MID statement)**

```

<Trace Level="MID">
<Time Millis="1298250694687"> 2011.02.21 06:41:34.687+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport.synchronization</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[Populating RECERTIFIER_DETAILS_INFO table ...]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.synchronization.AdhocSynchronizationManagerBean"
Method="populateDatabase"/>
<Thread>Default : 1</Thread>
</Trace>

```

- viii. The next step is to synchronize user access rights.

SynchronizationHelper.synchronizeUserRights() method populates all user tables for all entities passed as collection. It logs following debug statements for synchronization of each ACI class rights and user rights:

**(Converted DEBUG\_MAX to DEBUG\_MID statement)**

```

<Trace Level="MID">
<Time Millis="1298251803609"> 2011.02.21 07:00:03.609+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[INSERT INTO itimuser.classright_ADFeed (SET_ID, USERDN, USER_OWNER,
CONTAINERDN, RELATIONSHIP, RELATIONSHIP_SQL) VALUES (?, ?, ?, ?, ?, ?)]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.ACISynchronization" Method="getClassRightStmt"/>
<Thread>Default : 1</Thread>
</Trace>

```

```

<Trace Level="MID">
<Time Millis="1298251805906"> 2011.02.21 07:00:05.906+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[INSERT INTO itimuser.attrsright_ADFeed (erattrmapfilename, description,
erservicessomapping, erpersonsearchfilter, erevaluatesod, ernamingattribute, namingcontexts,
ernamingcontexts, owner, erparent, erpersonprofilename, erservicename, erprerequisite, erurl, eruid,
eruseworkflow, SET_ID) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.ACISynchronization" Method="getPrepStmtFinal"/>
<Thread>Default : 1</Thread>
</Trace>

```

ix. After creation of all entity tables, we need to create reporting views. Helper.createReportingViews() method creates the required views on Reporting Entity table.

J. This completes data synchronization process. Following debug statement is logged which shows total time required for data synchronization :

```

<Trace Level="MIN">
<Time Millis="1298253746765"> 2011.02.21 07:32:26.765+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport.synchronization</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[Synchronization completed in 7882141 milliseconds]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.synchronization.AdhocSynchronizationManagerBean"
Method="stageData"/>
<Thread>Default : 1</Thread>
</Trace>

```

K. It updates synchronization status in SYNCHRONIZATION\_HISTORY table as follows:

```

<Trace Level="MID">
<Time Millis="1298253746765"> 2011.02.21 07:32:26.765+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>

```

```

<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport.synchronization</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[UPDATE itimuser.SYNCHRONIZATION_HISTORY SET STATUS = ?,
    STATUS_DETAIL = ?, COMPLETED_TIME = 1298253746765 WHERE
    STARTED_TIME=1298245859484 AND SYNC_ID = 0 - with status = Success and status_detail
    = ]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.synchronization.AdhocSynchronizationManagerBean"
    Method="updateHistoryEntry"/>
<Thread>Default : 1</Thread>
</Trace>

```

- L. In AdhocSynchronizationManagerBean.releaseLock() method database lock is released after updating synchronization status. Here following debug statement is logged:

**(Added New DEBUG\_MID statement)**

```

<Trace Level="MID">
<Time Millis="1298253746765"> 2011.02.21 07:32:26.765+05:30</Time>
<Server Format="IP">itimwin3.tivlab.raleigh.ibm.com</Server>
<ProductId>CTGIM</ProductId>
<Component>com.ibm.itim.apps.ejb.adhocreport.synchronization</Component>
<ProductInstance>server1</ProductInstance>
<LogText><![CDATA[Released lock acquired for data synchronization after updating the status.]]></LogText>
<Source FileName="com.ibm.itim.apps.ejb.adhocreport.synchronization.AdhocSynchronizationManagerBean"
    Method="releaseLock"/>
<Thread>Default : 1</Thread>
</Trace>

```