Given tables with the following format, import the IP Addresses and Nodes into UCMDB.

OXYGEN.Integra...est - dbo.hosts

	Hostname	Host_id
▶	Test1	1
	Test2	2
	Test3	3
*	NULL	NULL

OXYGEN.Integra...dbo.Interfaces

	IPAddr	IntID
▶	10.1.1.1	1
	10.2.2.2	2
	10.3.3.2	3
	10.4.4.4	4
*	NULL	NULL

OXYGEN.Integr... - dbo.Mapping

	Host_ID	IF_ID
	1	1
	2	2
	2	3
▶	3	4
*	NULL	NULL

The primary key for the tables are the ID fields, and the primary key for the Mapping table is a combination of the Host_ID and IF_ID. There are foreign keys in this Mapping table, but they are used strictly for referential integrity of the data and are not required for the integration to function.

Steps to implement the GDBA:

1. Copy the generic DB adapter zip file onto your local computer (this is located on the UCMDB system under: <UCMDB_Install_Dir>\content\db-adapter.zip) and extract the files.
2. Under the adapterCode directory, change the folder name of 'GenericDBAdapter' to your integration name (in this example, test_integration).
3. Since we are adding IP Addresses, we will need to insert the UCMDB Probe Domain to the IP Address CIT. Open the <directory>\adapterCode\test_integration\META-INF\fixed_values.txt file and include the following line:
    a. entity[ip_address] attribute[routing_domain] value[${DefaultDomain}]
4. Edit the orm.xml file. The entities we will use in this example are node, ip_address and node_contains_ip_address. The following steps will be used to configure the orm.xml file:

5. The description should match the adapter name that is being used, so in our case, “test_integration”. The package should remain the “generic_db_adapter”.

```
<entity-mappings xmlns="http://java.sun.com/xml/ns/persistence/orm"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0"
            xsi:schemaLocation="http://java.sun.com/xml/ns/persistence/orm
http://java.sun.com/xml/ns/persistence/orm_1_0.xsd">
  <description>test_integration</description>
  <package>generic_db_adapter</package>
  <entity class="generic_db_adapter.node">
    <table name="hosts"/>
    <attributes>
      <id name="id1">
        <column updatable="false" insertable="false" name="Host_id"/>
        <generated-value strategy="TABLE"/>
      </id>
      <basic name="name">
        <column updatable="false" insertable="false" name="hostname"/>
      </basic>
    </attributes>
  </entity>
```

The entity class must be a CI Type that exists in UCMDB already. The table name is the table within the database that contains both an ID and the Host information. The ID attribute is required to identify specific hosts and will be used later in the mapping. We will be populating the ‘name’ attribute of this entity with the ‘hostname’ column in the hosts table.

6. For the next entity, we will be adding IP Addresses from the ‘interfaces’ table. The XML looks very similar:

```
  <entity name="ip_address" class="generic_db_adapter.ip_address">
    <table name="Interfaces"/>
    <attributes>
      <id name="id1">
        <column insertable="false" updatable="false" name="IntID"/>
        <generated-value strategy="TABLE"/>
      </id>
      <basic name="name">
        <column updatable="false" insertable="false" name="IPAddr"/>
      </basic>
    </attributes>
  </entity>
```

7. Next the link between the Node and the IP Address must be created. This will use the mapping table, and reference the IF_ID field (though it could reference both the Host_ID and IF_ID fields if desired).

```
  <entity name="node_containment_ip_address"
class="generic_db_adapter.node_containment_ip_address">
    <table name="Mapping"/>
    <attributes>
      <id name="id1">
        <column updatable="false" insertable="false" name="IF_id"/>
        <generated-value strategy="TABLE"/>
      </id>

      <many-to-one target-entity="node" name="end1">
        <join-column name="Host_ID"/>
      </many-to-one>
      <one-to-one target-entity="ip_address" name="end2">
        <join-column name="IF_ID"/>
      </one-to-one>
    </attributes>
  </entity>
```

The entity name for the container is in the format of [end1 CIT][link CIT][end2 CIT], so for a node containment ip_address, the value is: node_containment_ip_address, as is the class of generic_db_adapter.node_containment_ip_address. The ID is required in this block of code, and while this example works with a single ID of the Interface, both columns could be use referencing id1 and id2. The code for that would be:

```
<id name=”id1”>
        <column updatable=”false” insertable=”false” name=”IF_id”/>
        <generated-value strategy=”TABLE”/>
</id>
<id name=”id2”>
        <column updatable=”false” insertable=”false” name=”Node_id”/>
        <generated-value strategy=”TABLE”/>
</id>
```

The two ends of this link are ‘many-to-one’ and ‘one-to-one’, meaning each IP Address will be mapped to 1 node, but a node may be mapped to many IP Addresses. The columns included are from the table “Mapping” and reference the Hosts and Interfaces tables. Host_ID is of type ‘Node’ and references the Hostname created in entity one and IF_ID is the interface identifier which references the values in the Interfaces table. The join is done for you by the UCMDB mapping.

8. Once the orm.xml file is edited, the next step is to edit the discoveryPatterns\db_adapter.xml file. The name of the file should be changed to reflect the integration, such as

'test_integration.xml'. The id parameter should also be changed to 'test_integration' as follows:

```
<pattern id="test_integration" xsi:noNamespaceSchemaLocation="../../Patterns.xsd"
description="Uses the GDB-Framework to Populate and Federate data from a SMS database"
schemaVersion="9.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
displayName="Test_Integration">
```

If population is to be used, make sure the 'support-replicatioin-data' block is set up as below:

```
<support-replicatioin-data>
    <source>
        <changes-source>
            <!--<changes-source/>-->
        </changes-source>
    </source>
</support-replicatioin-data>
```

Add the 'adapterTemplate' section to reference the correct name of the adapters, and the correct query (which will be discussed in a few minutes). This should be inserted after </adapterInfo>.

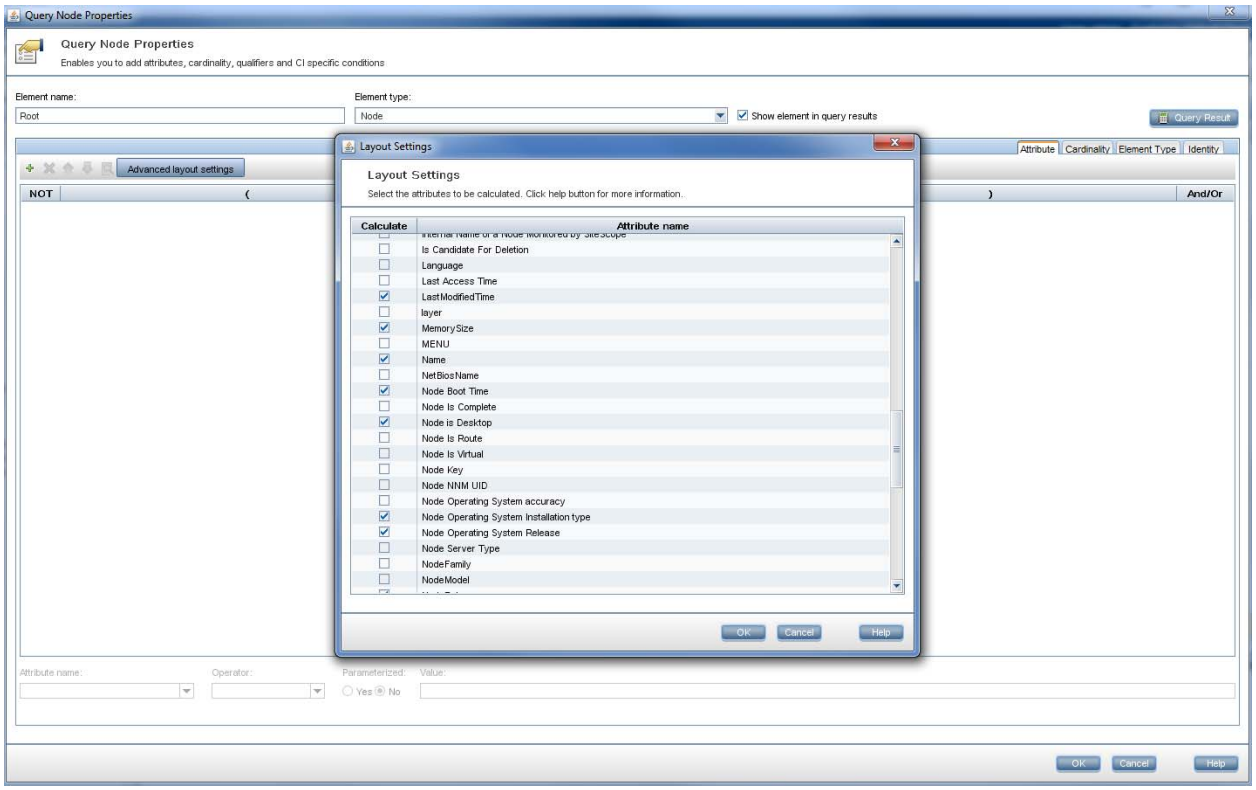
```
<adapterTemplates>
    <adapterTemplate  name="test_integration" description="test_integration">
        <destination-config destination-id="test_integration">
            <adapter-id>test_integration</adapter-id>
            <destination-description/>
            <classes/>
            <type-values>
                <type-value name="dbtype" type="string">SQLServer</type-value>
                <type-value name="dbname" type="string">Integration_test</type-
value>
            </type-values>
        </destination-config>
        <population-jobs>
            <data-acquisition-population-job>
                <unit-id>Test_Integration Population job</unit-id>
                <query-names>
                    <query-selection>
                        <query-name>hostDataFromIntegration</query-name>
                        <allow-deletion>false</allow-deletion>
                    </query-selection>
                </query-names>
                <allow-deletion>false</allow-deletion>
            </data-acquisition-population-job>
```

```
            </population-jobs>
        </adapterTemplate>
</adapterTemplates>
```

Change the category from Generic to something else so it will show up in the integrations list when creating a new adapter.

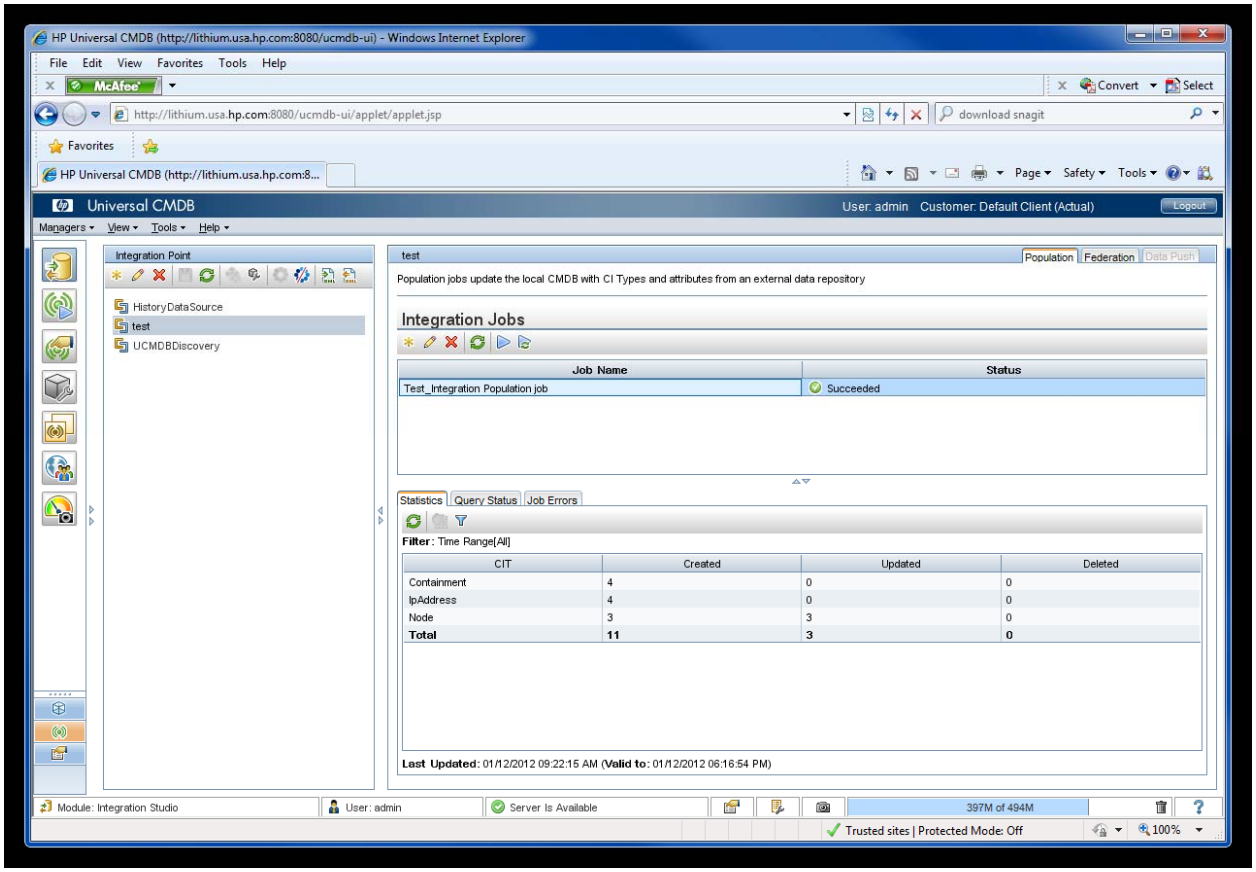
```
<category>test</category>
```

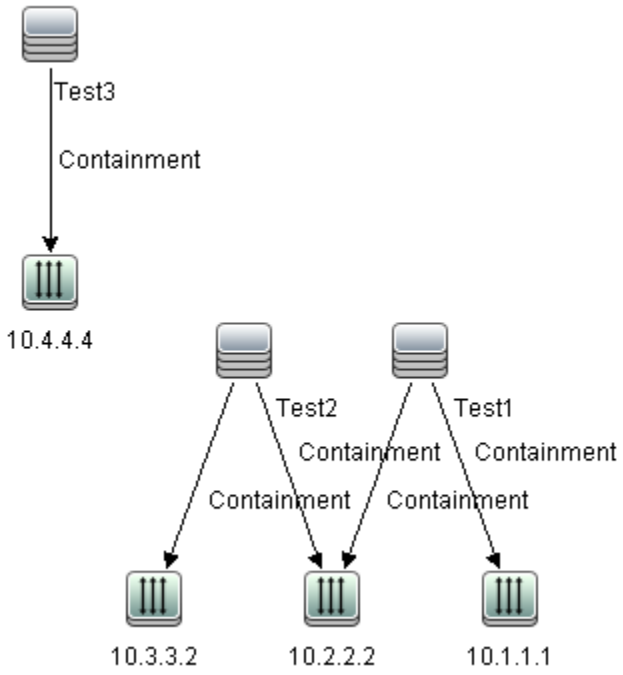
The dbname field is the database name from the MSSQL server (in our example) that contains the tables for importing data from. This can be overridden by the integration point, as well as the server type. For Oracle RAC, see the UCMDB documentation to configure the parameters section on page 140 of the Developer's Reference Manual. The query name that is referenced (in our case, the hostDataFromIntegration was copied in the modeling studio from the SMSDataFromIntegration. The fields that will be updated should be exposed via the advanced layout settings in the Query Node Properties screen, such as the following image:



9. The next thing to do is to remove the jar files from the adapter (if they exist). These would be the asm.jar, asm-attrs.jar, cglib.jar, db-adapter.jar, jboss-archive-browser.jar and saxon-b.jar. Then a new zip file should be created containing the 'adapterCode' and 'discoveryPatterns' folders at the top level. The undeployResources directory may be removed.
10. Deploy the zip file to the UCMDB Server via package manager.

11. Copy the Integration-> SMS Sync -> hostDataFromSMS query to hostDataFromIntegration query (specified in the test_integration.xml file) and remove the ‘interface’ CI (we aren’t adding that one).
12. Add the integration in the ‘Data Flow Management -> Integration Studio and provide the database and credentials.
13. Run the Integration.





The links between Test2 and Test1 to the 2nd IP are incorrect.