Peregrine

Reporting Data Store Administration Guide

For Windows

PART NO: DBNI-51-EN05

Peregrine SYSTEMS®

The information in this document is subject to change without notice and does not represent a commitment on the part of Peregrine Systems, Inc. Contact Peregrine Systems, Inc., Customer Support to verify the date of the latest version of this document. The names of companies and individuals used in the sample database and in examples in the manuals are fictitious and are intended to illustrate the use of the software. Any resemblance to actual companies or individuals, whether past or present, is purely coincidental. If you need technical support for this product, or would like to request documentation for a product for which you are licensed, contact Peregrine Systems, Inc. Customer Support by email at support@peregrine.com. If you have comments or suggestions about this documentation, contact Peregrine Systems, Inc. Technical Publications by email at doc_comments@peregrine.com. This edition of the document applies to version 5.1 of the licensed program.

Peregrine Systems, Inc.
Worldwide Corporate Headquarters
3611 Valley Centre Drive San Diego, CA 92130
Tel 800.638.5231 or 858.481.5000
Fax 858.481.1751
www.peregrine.com

Contents

About this Guide

This document is for advanced users who are going to customize the Reporting Data Store (RDS). If you intend to customize the RDS_etl.xml file, contact Peregrine Educational Services about BI Portal training and engage Peregrine Professional Services for additional assistance. This *Reporting Data Store Administration Guide* provides information about customizing the RDS.

Using this Guide

This guide includes the following chapters.

This chapter	Describes
Chapter 1	Provides an overview of the Reporting Data Store (RDS).
Chapter 2	Explains how to utilize the Connect-It tools to perform RDS data synchronization and how to tailor the scenario for your deployment.
Chapter 3	Discusses the recommended workflow for creating useful and effective RDS customizations.
Chapter 4	Discusses RDS schema customization and ServiceCenter triggers customization.
Chapter 5	Discusses how to customize the RDS Connect-It scenario, rds_sc.scn, for ServiceCenter.
Chapter 6	Explains how to use the Business Objects Designer tool to customize a universe file.

Prior to using this guide, you should read the following sections:

- *Related Documentation*
- *Documentation Conventions* on page 9
- *Contacting Peregrine Systems* on page 9

Related Documentation

In addition to this guide, the following documentation is available for the RDS product:

This manual…	Provides information about…
BI Portal User Guide	standard reports and describes how to create and work with both standard and custom reports.
BI Portal Installation Guide	Installing and configuring the application and database servers for the RDS.

This manual…	Provides information about…
BI Portal Release Notes	Last-minute enhancements, known issues, and closed issues.
WebIntelligence User's Guide	Describes how to use WebIntelligence (a component of Business Objects) for building and running queries, reporting, and analysis.

Documentation Conventions

The following typographical conventions are used in this guide.

Text Formatting	Meaning
italics	Text that acts as a placeholder for information you will provide. Italics are also used for book titles and for emphasis.
sans serif font	Text that you type. Examples are filenames and URLs. This font is also used for samples of code and commands.
bold	Names of user interface elements. Examples are menu items and names (select **Open** from the **File** menu), button names (click **Accept**), and names of screens or dialogs (the **Server Manager** window).

Contacting Peregrine Systems

For help with this release, you can contact customer support, download documentation or schedule training.

Customer Support

For further information and assistance with Product Name Short in general, contact Peregrine Systems' Customer Support at the Peregrine CenterPoint web site.

To contact customer support:

1. In a browser, navigate to http://support.peregrine.com
2. Log in with your user name and password.
3. Follow the directions on the site to find the information you need.

The KnowledgeBase contains informational articles about all categories of Peregrine products. If the KnowledgeBase does not contain an article that addresses your concerns, you can search for information by product; search discussion forums; and search for product downloads.

Documentation web site

For a complete listing of current Reporting Data Store documentation, see the Documentation pages of the Peregrine Customer Support web site.

To view the document listing:

1. In a browser, navigate to http://support.peregrine.com.
2. Log in with your login user name and password.
3. Click either **Documentation** or **Release Notes** at the top of the page.
4. Click the Reporting Data Store link.
5. Click a product version link to display a list of documents that are available for that version of Reporting Data Store.
6. Documents may be available in multiple languages. Click the Download button to download the PDF file in the language you prefer.

You can view PDF files using Acrobat Reader, which is available on the Customer Support Web site and through Adobe at http://www.adobe.com.

Important: Release Notes for this product are continually updated after each release of the product. Ensure that you have the most current version of the Release Notes.

Education Services web site

Peregrine Systems offers classroom training anywhere in the world, as well as "at your desk" training via the Internet. For a complete listing of Peregrine's training courses, refer to the following web site:

http://www.peregrine.com/education

You can also call Peregrine Education Services at +1 858.794.5009.

1 Reporting Data Store and Schema

CHAPTER

This chapter provides an overview of the Reporting Data Store (RDS) and serves as an introduction to some of the RDS design concepts you need to be familiar with if you intend to do any customizing of the out-of-box RDS used with BI Portal. This chapter discusses the following topics:

Note: The information in this section provides an introduction to the Reporting Data Store (RDS). If you intend to customize the RDS_etl.xml file, you should contact Peregrine Educational Services about BI Portal training and engage Peregrine Professional Services for additional assistance.

What is the Reporting Data Store (RDS)

The RDS is a repository of integrated information that is available for queries and analysis and includes Extract, Transform and Loading (ETL) processes and metadata.

The RDS provides the following advantages:

- Relieves reporting pressure on the database.

- Enables administrators to restructure data to speed up data analysis and reporting capabilities.
- Reduces the difficulty users have with generating new reports.
- Enables administrators to create "clean data" that does not require wholesale changes to the transactional system (ServiceCenter) or business processes.
- Enables hierarchy analysis.
- Allows historical analysis by providing a data source supporting a longer span of time.

RDS design overview

The RDS uses Dimensional Modeling to present data in a standard framework that is intuitive and allows for high performance access. It is gracefully extendable and provides easy management of aggregates.

The RDS defines tables based on the following categories:

- Dimension tables
- Fact tables
- Associate or join tables and bridge tables
- Direct mapping tables from ServiceCenter (SC) transaction tables
- Aggregates of fact tables
- Hierarchy tables
- RDS system tables

Dimension tables

Most dimension table fields are directly mapped from SC tables through Connect-It. Dimension tables add additional RDS ETL (Extract, Transform and Loading) attributes. ServiceCenter tables should be considered as possible dimension tables if they include historical, reserved data or time associated data or the table is used as search criteria for multiple ServiceCenter reporting modules.

Each dimension table has the system key, Z_RDS(*tablename*)ID. The surrogate keys allow the data warehouse to assign a new key version for Slowly Changing Dimensions and allow the data warehouse to encode uncertain, not known, not recorded, and null record types. Every join between dimension and fact tables should be based on surrogate keys.

Fact tables

Fact tables are the RDS tables created for measurements for the associated dimension tables used by the reporting modules.

Associate or join tables and bridge tables

Associate or bridge tables created in the RDS are used for sub-dimension tables or solving many-to-many relationships.

Direct mapping tables from ServiceCenter transaction tables

BI Portal uses ServiceCenter transaction tables for detail reporting and as search criteria for one ServiceCenter module. These tables are directly mapped from ServiceCenter tables into the RDS.

Aggregates of fact tables

Measures grouped by common sort fields are pre-aggregated to allow faster summarization of data.

Hierarchy tables

Hierarchy tables contain fields create parent-child relationships between records. For example, manager is the parent field name in the table, *Contact*. The maximum hierarchy level is 10.

RDS system tables

In the RDS, there are system tables defined to track RDS schema metadata, RDS user security, the ETL process status and time dimension. All the system table names begin with 'RDS_'; with only one exception - BIRECORDDELETE.

RDS naming conventions

The table naming conventions are:

- Dimension table names end with "_D" and have a maximum length of eighteen characters. The primary key for dimension table is Z_RDSXXX_DID. Sample name is XXX_D.

- The dimension table names use the first eight characters from original source table names without the “S”.
- Fact table names end with “_F”.
- Do not use count character for measurement names for fact tables.
- Aggregate table names end with “_AGG”.
- Associate or bridge table names use the first eight characters from the two associated table names with an underscore (_) as the separator.
- All the direct mapping table names use the same name as the source name.
- Dots, (.), in ServiceCenter field names should be converted to underscores for RDS field names. For example, convert parent.name to parent_name.
- Direct Mapping Indexes length can be no longer than 18 characters.
- The RDS system table names start with “RDS_”.

RDS metadata overview

The RDS metadata consists of the RDS schema, ETL process and RDS semantic layer, and the RDS universe.

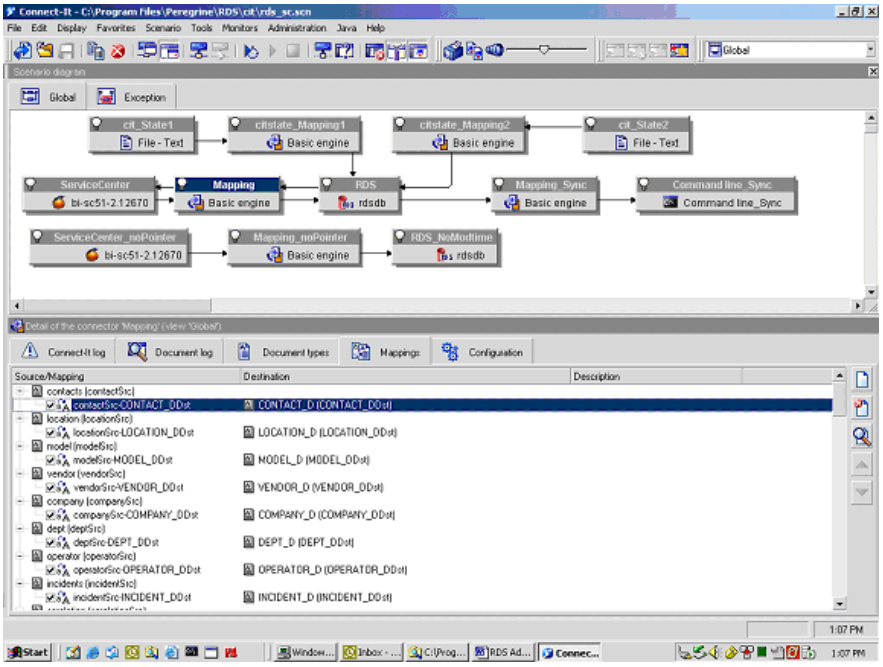
RDS schema

The RDS schema data is provided by **rds_etl.xml**. The XML schema syntax is defined by **rds_etl.xsd**. All these files are stored in **\conf** folder of the RDS installation directory.

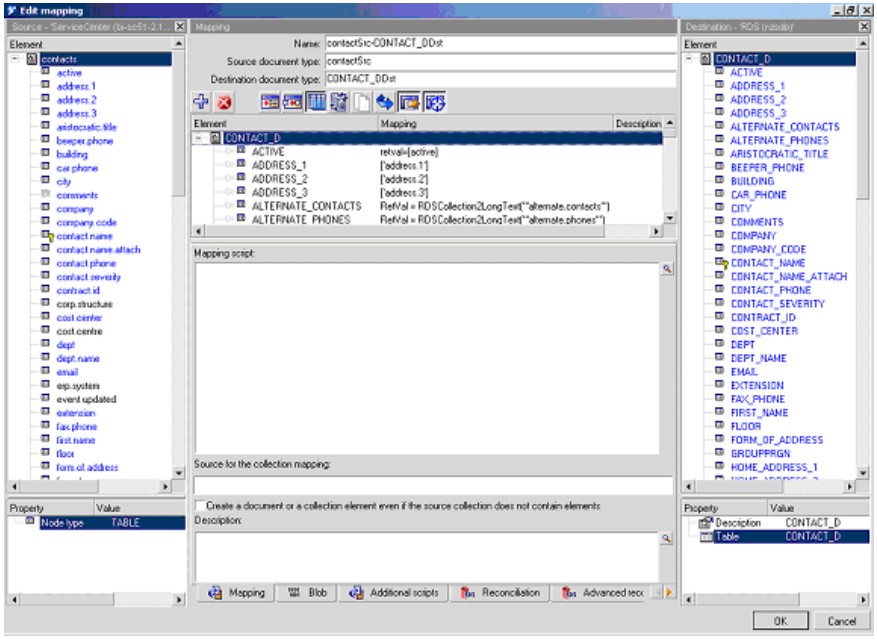
RDS ETL mapping information

The metadata for the extraction, transformation and mapping is described by Connect-It RDS scenario (**rds_sc.scn**). You can launch Connect-It scenario builder and open **rds_sc.scn** from **\cit** folder of the RDS installation directory. You can use the Edit function to view any Source/Mapping and review the ETL mapping information.

The following diagram shows that the mapping for the *contacts* table.

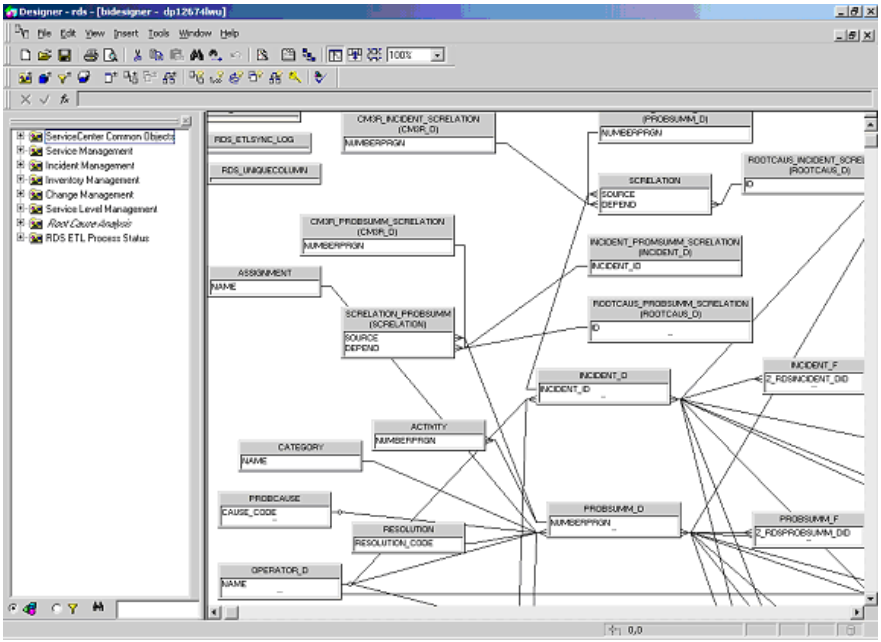


After you click the **edit a mapping** button, the Edit Mapping window opens. The ServiceCenter data source appears in the left pane and the right pane shows the RDS destination table. In the middle pane, the mapping script is defined for each attribute. The *Connect-It User Guide* provides more information about how to use this tool.

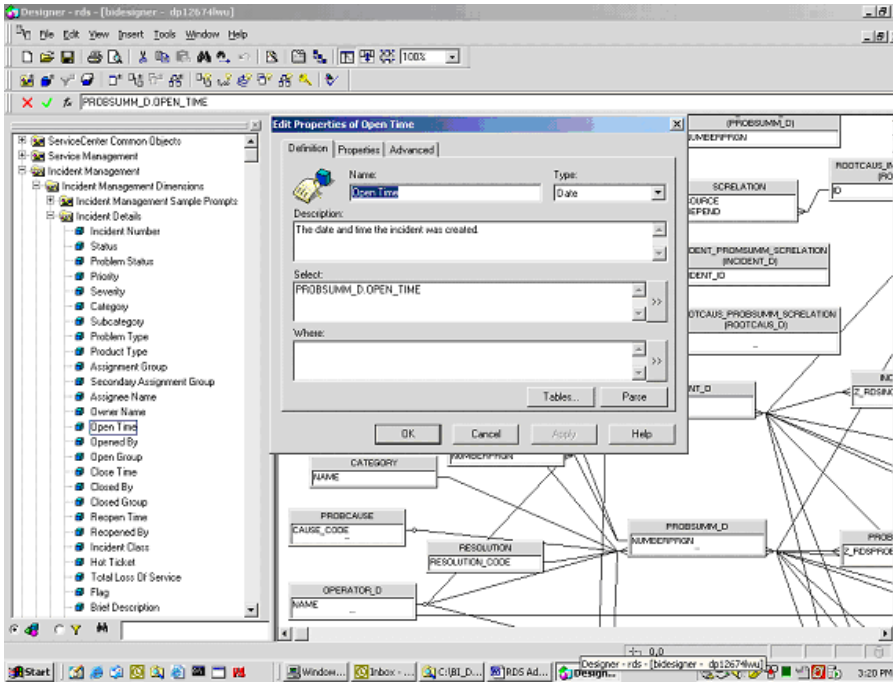


RDS semantic layer - universe

The Business Objects (BO) designer tool is used to develop the RDS universe. The diagram below shows the RDS universe displayed in the designer tool.



You can use Business Objects designer tool to view the object definitions, which you can use to determine how to map the fields of RDS tables. To open an object's properties select the section. The following figure shows an example of how the tools displays the properties.



You can use the Business Objects designer tool to produce a PDF file of the universe metadata. The **rds.unv** file is supplied in the **Support** folder on the BI Portal installation CD. When you open this file using the Business Objects designer tool, you can save it as a PDF document.

RDS schema file description

The RDS database is created based on the schema defined in file, **rds_etl.xml**. This schema is built based on the XML schema definition defined in **rds_etl.xsd**. These two files normally reside in the directory, **\conf**, under the RDS installation root directory. The detail of the **rds_etl.xsd** is included in the appendix.

The following definitions organize the RDS schema in sequential order:

- A list of dimension table definitions
- A list of associate table definitions
- A list of hierarchy table definitions
- A list of direct mapping table definitions

The RDS schema uses the following XML syntax:

```
<rds>
  <dimensions/>
  <associates/>
  <hierarchies/>
  <directmappings/>
</rds>
```

The following sections list the XML schema definition and provide examples each for the table types.

Dimension table schema definition

Dimension tables are grouped under a root element, **dimensions**. Under this root element, there can be one or more dimension table definitions. Each dimension table definition is tagged under an element name, **dimension**. In general, a dimension table definition contain the following:

- A dimension table name
- Required elements
  - A data source table name
  - A list of field definitions
  - A list of unique key definitions
- Optional elements
  - A list of fact table definitions
  - A list of SCD key definitions
  - A list of aggregate key definitions

For detail XML schema for this table type, examine the definition section for a complex type of DimensionType in **rds_etl.xsd**.

The following example shows the RDS schema definition for device table in ServiceCenter:

```
<!--  Device inventory dimension ETL definition  -->
  <dimension name="DEVICE" >
   <dimensionTableName>DEVICE_D</dimensionTableName>
   <dataSourceTableName>device</dataSourceTableName>
   <dimensionTableFields>
    <dimensionTableField name="logical_name" type="char" size="60"/>
    <dimensionTableField name="vendor" type="char" size="60"/>
    . . .
    <dimensionTableField name="comments" type="long"/>
    <!-- !!! SC 5.1 renamed attributes -->
    <dimensionTableField name="cost_center" type="char" size="60"/>
    <!-- !!! SC 5.1 new attributes -->
    <dimensionTableField name="assignment" type="char" size="60"/>
    <dimensionTableField name="port_desc" type="long"/>
    . . .
   </dimensionTableFields>
   <uniqueKeys name="device_unique">
    <uniqueKey fieldName="logical_name" srcFieldName=""'logical.name'""
            srctype="char" size="60" defaultValue="no match"/>
   </uniqueKeys>
   <facts>
    <fact name="DEVICE_F">
     <factKeys>
      <factKey name="Z_RDSDEVICE_DID"/>
      <factKey name="Z_RDSCONTACT_DID" fieldName="CONTACT_NAME"
            tableName="CONTACT_D" matchFieldName="CONTACT_NAME"/>
      <factKey name="Z_RDSLOCATION_DID" fieldName="LOCATION"
            tableName="LOCATION_D" matchFieldName="LOCATION"/>
      <factKey name="Z_RDSMODEL_DID" fieldName="part_no"
            tableName="MODEL_D" matchFieldName="part_no"/>
      <factKey name="Z_RDSVENDOR_DID" fieldName="VENDOR"
            tableName="VENDOR_D" matchFieldName="VENDOR"/>
      <factKey name="z_rdscompany_did" fieldName="COMPANY"
            tableName="COMPANY_D" matchFieldName="COMPANY"/>
      <factKey name="z_rdsdept_did" fieldName="dept" tableName="DEPT_D"
            matchFieldName="DEPT"/>
      <factKey name="install_dateID" fieldName="install_date"
```

```
            tableName="RDS_TIMEDIM_D" matchFieldName="FULLDATE"/>
        <factKey name="last_scanID" fieldName="last_scan"
            tableName="RDS_TIMEDIM_D" matchFieldName="FULLDATE"/>
      </factKeys>
      <factMeasures>
        <!-- !!! Don't use count in measurement names if it is possible, because BO E6
            bugs !!! -->
        <factMeasure name="pcount" srcType="src"/>
        <factMeasure name="total_downtime" srcType="src"/>
        <factMeasure name="breaks" srcType="src"/>
        <factMeasure name="DEVICE_number" srcType="rdsfactless"/>
        <factMeasure name="DEVICE_SCAN" srcType="rds"
            sqlConditions="LAST_SCAN is not null" resetCount=""/>
      </factMeasures>
    </fact>
  </facts>
  <scdKeys>
    <scdKey rdsFieldName="network_name" srcFieldName=""'network.name'""
        srctype="char"/>
    <scdKey rdsFieldName="domain" srcFieldName="domain" srctype="char"/>
    <scdKey rdsFieldName="contact_name" srcFieldName=""'contact.name'""
        srctype="char"/>
  </scdKeys>
  <aggregateKeys>
    <aggregateKey name="type" type="char" size="60"/>
    <aggregateKey name="subtype" type="char" size="60"/>
    <aggregateKey name="Istatus" type="char" size="60"/>
    <aggregateKey name="vendor" type="char" size="60"/>
    <aggregateKey name="install_date" type="TIME"/>
  </aggregateKeys>
</dimension>
```

Fact table schema definition

Fact tables are defined under a root element, **facts**. Under this root element, one or more fact tables can be defined. Each fact table is tagged by an element name, **fact**.

The following shows the XML schema definition for the **FactType** table type:

```
<xsd:complexType name="FactType">
    <xsd:sequence>
    <xsd:element name="factKeys">
      <xsd:complexType>
        <xsd:sequence>
        <xsd:element name="factKey" type="FactKeyType" minOccurs="1"
                maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="factMeasures">
     <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="factMeasure" type="FactMeasureType" minOccurs="1"
           maxOccurs="unbounded"/>
        </xsd:sequence>
     </xsd:complexType>
    </xsd:element>
     </xsd:sequence>
     <!-- the name has to be same as dimension name before '_' -->
   <xsd:attribute name="name" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="FactKeyType">
   <xsd:attribute name="name" type="xsd:string" use="required"/>
   <xsd:attribute name="fieldName" type="xsd:string"/>
   <xsd:attribute name="tableName" type="xsd:string"/>
   <xsd:attribute name="matchFieldName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="FactMeasureType">
    <!-- Don't use count in measurement names, because BO E6 bugs  -->
    <!-- only ONE rdsfactless measurement, ex open ticket count -->
    <!-- only ONE rds measurement with resetCount for rdsfactless not blank, ex close
         ticket count -->
   <xsd:attribute name="name" type="xsd:string" use="required"/>
   <xsd:attribute name="srcType" type="xsd:string" use="required"/>
   <xsd:attribute name="sqlConditions" type="xsd:string"/>
   <xsd:attribute name="resetCount" type="xsd:string"/>
</xsd:complexType>
```

The following is an example of a fact table definition in dimension table DEVICE:

```
<facts>
    <fact name="DEVICE_F">
     <factKeys>
       <factKey name="Z_RDSDEVICE_DID"/>
       <factKey name="Z_RDSCONTACT_DID" fieldName="CONTACT_NAME"
            tableName="CONTACT_D" matchFieldName="CONTACT_NAME"/>
       <factKey name="Z_RDSLOCATION_DID" fieldName="LOCATION"
            tableName="LOCATION_D" matchFieldName="LOCATION"/>
       <factKey name="Z_RDSMODEL_DID" fieldName="part_no"
            tableName="MODEL_D" matchFieldName="part_no"/>
       <factKey name="Z_RDSVENDOR_DID" fieldName="VENDOR"
            tableName="VENDOR_D" matchFieldName="VENDOR"/>
       <factKey name="z_rdscompany_did" fieldName="COMPANY"
            tableName="COMPANY_D" matchFieldName="COMPANY"/>
       <factKey name="z_rdsdept_did" fieldName="dept" tableName="DEPT_D"
            matchFieldName="DEPT"/>
       <factKey name="install_dateID" fieldName="install_date"
            tableName="RDS_TIMEDIM_D" matchFieldName="FULLDATE"/>
       <factKey name="last_scanID" fieldName="last_scan"
            tableName="RDS_TIMEDIM_D" matchFieldName="FULLDATE"/>
     </factKeys>
     <factMeasures>
       <!-- !!! Don't use count in measurement names if it is possible, because BO E6
            bugs !!! -->
       <factMeasure name="pcount" srcType="src"/>
       <factMeasure name="total_downtime" srcType="src"/>
       <factMeasure name="breaks" srcType="src"/>
       <factMeasure name="DEVICE_number" srcType="rdsfactless"/>
       <factMeasure name="DEVICE_SCAN" srcType="rds"
            sqlConditions="LAST_SCAN is not null" resetCount=""/>
     </factMeasures>
    </fact>
</facts>
```

Associate table schema definition

Associate tables are defined under a root element, associates. Each associate table is tagged by an element, **associate**. Under this root element there can be one or more associate table definition. An associate table contains the following attributes:

- Require attributes
  - Name of the table
  - Table name and field name for table 1 of the association relationship
  - Table name and field name for table 2 of the association relationship
- Optional attributes
  - Name of the long field name
  - RDS version string

The following shows the XML schema definition for an associate table:

```
<xsd:complexType name="AssociateType">
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="rdsVersion" type="xsd:string" use="optional"/>
    <xsd:attribute name="tableName" type="xsd:string" use="required"/>
    <xsd:attribute name="fieldName" type="xsd:string" use="required"/>
    <xsd:attribute name="longFieldName" type="xsd:string"/>
    <xsd:attribute name="tableName2" type="xsd:string" use="required"/>
    <xsd:attribute name="fieldName2" type="xsd:string" use="required"/>
</xsd:complexType>
```

The following is an example of an associate table definition in RDS schema for the ServiceCenter *device* table:

```
<associates>
  <associate name="INCIDENT_MODEL_ASS" rdsVersion="5.0"
      tableName="INCIDENT"
      fieldName="MODEL" tableName2="MODEL" fieldName2="MODEL"/>
  <associate name="PROBSUMM_MODEL_ASS" tableName="PROBSUMM"
      fieldName="MODEL"
      tableName2="MODEL" fieldName2="MODEL"/>
  <associate name="CM3R_DEVICE_ASS" tableName="CM3R"
      fieldName="logical_name"
      longFieldName="assets" tableName2="DEVICE"
```

```
        fieldName2="LOGICAL_NAME"/>
</associates>
```

Hierarchy table schema definition

Hierarchy tables are defined under a root element, hierarchies. Each hierarchy table definition is tagged by an element name, hierarchy. This element can contain one of more hierarchy tables. A hierarchy table has the following attributes:

- Required attributes
  - Name of the table
  - Type string
  - Level string
  - Unique field name
- Optional attributes
  - RDS version string
  - Parent table name and parent field name
  - Full path field name

The following shows the XML schema definition for a hierarchy table:

```
<xsd:complexType name="HierarchyType">
        <xsd:attribute name="tablename" type="xsd:string" use="required"/>
        <xsd:attribute name="rdsVersion" type="xsd:string" use="optional"/>
        <xsd:attribute name="type" type="xsd:string" use="required"/>
        <xsd:attribute name="level" type="xsd:string" use="required"/>
        <xsd:attribute name="uniquefieldName" type="xsd:string" use="required"/>
        <xsd:attribute name="parenttableName" type="xsd:string"/>
        <xsd:attribute name="parentfieldname" type="xsd:string"/>
        <xsd:attribute name="fullpathfieldname" type="xsd:string"/>
</xsd:complexType>
```

The following is an example of the hierarchy table type:

```
<hierarchies>
        <hierarchy tablename="location" rdsVersion="5.0"
                type="fullpath" level="5" uniquefieldName="location"
                fullpathfieldname="location_full_name"/>
```

```
</hierarchies>
```

Direct mapping table schema definition

Direct mapping tables are defined under a root element, **directMappings**. Under this root element there can be one or multiple direct mapping table definitions. Each direct mapping table is tagged by an element name, **directMapping**.

The following is an XML schema definition for **directMapping** table type:

```
<xsd:element name="directMapping">
        <!-- If dataSourceTableName is defined, SC delete eventout put can be used
            to delete sync -->
        <!-- parentDimensionTableName can be multiple src table names separate
            by | -->
        <!-- If parentDimensionTableName is not null, DEL type index has to be
            defined -->
       <xsd:complexType>
            <xsd:sequence>
                 <xsd:element ref="directMappingFields"/>
                 <xsd:element ref="directMappingIndexes"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string" use="required"/>
            <xsd:attribute name="rdsVersion" type="xsd:string"/>
            <xsd:attribute name="dataSourceTableName" type="xsd:string"/>
            <xsd:attribute name="parentDimensionTableName" type="xsd:string"/>
       </xsd:complexType>
</xsd:element>
```

This XML schema references two additional definitions: **directMappingFields** and **directMappingIndexes**. For additional details for these two definitions, examine the appropriate sections of **RDS_etl.xsd**.

The following is an example of a direct mapping table for the ServiceCenter *assignment* table:

```
<directMapping name="assignment" dataSourceTableName="assignment">
   <directMappingFields>
    <directMappingField name="name" type="char" size="60"/>
    <directMappingField name="assignment2" type="char" size="60"/>
```

```
 <directMappingField name="assignment3" type="char" size="60"/>
 <directMappingField name="reassignment" type="char" size="60"/>
 <directMappingField name="calendar_name" type="char" size="60"/>
 <directMappingField name="duty_hours" type="char" size="30"/>
 <directMappingField name="type" type="char" size="30"/>
 <directMappingField name="reassign" type="char" size="1"/>
 <directMappingField name="route_to" type="float"/>
 <directMappingField name="route_if" type="char" size="60"/>
 <directMappingField name="wdManagerName" type="char" size="60"/>
 <directMappingField name="sysmodcount" type="float"/>
 <directMappingField name="sysmoduser" type="char" size="30"/>
 <directMappingField name="sysmodtime" type="date"/>
 <directMappingField name="company" type="char" size="60"/>
 <directMappingField name="operators" type="long"/>
</directMappingFields>
<directMappingIndexes>
 <directMappingIndex name="UNIQASSIGNMENT_IDX">
  <DirectMappingIndexKey fieldName="name" srcFieldName="name"
        srctype="char" size="60" seqIndex="1"/>
 </directMappingIndex>
</directMappingIndexes>
</directMapping>
```

Slowly changing dimension (SCD) keys schema definition

The SCD keys are grouped under a root element, **scdKeys**. Each SCD key definition is tagged by **scdKey** and has the following attributes:

- rds field name
- source field name
- source type

The “Slowly Changing Dimension” data problem is a common database problem, particularly for data warehousing. Essentially, this situation applies to cases where the attribute for a record varies over time.

There are, in general, three ways to solve this type of problem, and they are categorized as follows:

Type 1: The new record replaces the original record. No trace of the old record exists.

Type 2: A new record is added to the table to represent the new information. And the original record is marked as non-active.

Type 3: The original record is modified to reflect the change.

The RDS only uses type 2 - create a new dimension record. These are used to partition the historical data and maintain historical accuracy of the RDS.

The following shows the XML schema definition for a SCD key:

```
<xsd:complexType name="scdKeyType">
   <xsd:attribute name="rdsFieldName" type="xsd:string" use="required"/>
   <!-- !!!  src field name needs '' for the sc field name with dot. ex ""'alert.status'"" -->
     <xsd:attribute name="srcFieldName" type="xsd:string" use="required"/>
     <xsd:attribute name="srctype" type="xsd:string" use="required"/>
</xsd:complexType>
```

The following XML is an example of **scdKeys** defined in the schema for the ServiceCenter *device* table in the RDS database.

```
<scdKeys>
    <scdKey rdsFieldName="network_name" srcFieldName=""'network.name'""
       srctype="char"/>
    <scdKey rdsFieldName="domain" srcFieldName="domain" srctype="char"/>
    <scdKey rdsFieldName="contact_name" srcFieldName=""'contact.name'""
       srctype="char"/>
</scdKeys>
```

Aggregate keys schema definition

The aggregate keys are grouped under a root element, **aggregateKeys**. Each aggregate key definition is tagged by an element name, **aggregateKey**.

The following shows the XML schema definition for an **aggregateKey**.

```
<xsd:complexType name="aggregateKeyType">
        <!-- only ONE and LAST aggregate field is used as the type of TIME -->
       <xsd:attribute name="name" type="xsd:string" use="required"/>
       <xsd:attribute name="type" type="xsd:string" use="required"/>
       <xsd:attribute name="size" type="xsd:string"/>
</xsd:complexType>
```

The following is an example of aggregate keys defined in the schema for the ServiceCenter *device* table:

```
<aggregateKeys>
    <aggregateKey name="type" type="char" size="60"/>
    <aggregateKey name="subtype" type="char" size="60"/>
    <aggregateKey name="Istatus" type="char" size="60"/>
    <aggregateKey name="vendor" type="char" size="60"/>
    <aggregateKey name="install_date" type="TIME"/>
</aggregateKeys>
```

RDS dimension attributes

All the dimension keys' value is 2 if it is not the match type and 1 if its value is blank.

For updates, the dimensions attributes provide the following information, which is used to track changes to RDS table reporting:

- When new records are added into rds dimension tables (XXX_D), Z_RDSCREATEDDATE is set to data update sync time.
- When the records are updated into rds dimension tables (XXX_D), Z_RDSLASTMODDATE is set to data update sync time.
- When the records are deleted from SC, attributes are updated as:
  - Z_RDSACTIVESTATUSIND = 'N'.
  - Z_RDSDELETEDDATE is set to the timestamp value at the time the record was deleted in ServiceCenter.
  - Z_RDSTRANSENDDATE is set to data update sync time.
  - Z_RDSLASTMODDATE is set to data update sync time.
- When the records are updated as SCD, the original dimension table record attributes are updated as:
  - Z_RDSTRANSLASTIND= 'N'.
  - Z_RDSLASTMODDATE is set to the data update sync time.
  - Z_RDSTRANSENDDATE is set to the data update sync time.
  - And the new record is added as ADD case.

- Z_RDSTRANSLASTIND = 'N' or Z_RDSACTIVESTATUSIND = 'N' means the dimension record is not active any more.
- Z_RDSOBSOLETEIND = 'Y' means the fact record is either a historic or a deleted record.

RDS InitRun command

You use the RDSInitRun command to have the changes you made in the RDS_etl.xml file take effect.

Usage	RDSInitRun RDS-Schema-XML-File	Modification-Indicator
Sample:	RDSInitRun (the default xml file is set as rds_etl.xml	
Sample:	RDSInitRun rds_etl.xml	
Sample:	RDSInitRun testdirect.xml	m
Sample:	RDSInitRun testdim.xml	m
Sample:	RDSInitRun rds_etl.xml	time
Sample:	RDSInitRun rds_etl.xml	m_upgrade

In the sample commands, **m** means modified.

Synchronizing record deletions in ServiceCenter with the RDS

For BI Portal 5.0, in order to synchronize records deleted from the ServiceCenter database with those in the RDS, BI Portal performed whole table scans for marking records for deletion in RDS. It is a very costly operation. For BIP 5.1, we implemented a new mechanism to improve the general performance for marking records for deletion in RDS.

BI Portal uses the following mechanisms to synchronize record deletions in the ServiceCenter database with the RDS:

- When records of a set of pre-defined ServiceCenter tables are being deleted, the associated table-level post-delete trigger is invoked. The trigger application saves the deletion events in ServiceCenter. One deletion event is saved for each deleted record. In each of these events, the system saves the table name, primary key(s) of the specified deleted record, and the timestamp of the deletion.

- When the next RDS data synchronization cycle starts, the RDS ETL process retrieves these deletion events from ServiceCenter. The data recorded in each event is processed in the following manner:
  - Find the RDS table corresponding to the specified ServiceCenter table.
  - Select the deleted record based on the primary key(s).
- Mark the status of the record as Inactive, update the deletion timestamp of the record, and update the record accordingly.
- After the RDS ETL process completes, all the records that were deleted from ServiceCenter are properly synchronized in the RDS. The event records in ServiceCenter for the deletions are removed.

CHAPTER 2 Using Connect-It to Handle RDS Data Synchronization

This chapter explains how to utilize the Connect-It (CIT) tools to perform Reporting Data Store (RDS) data synchronization and how to tailor the scenario for your deployment. This chapter discusses the following topics

RDS data synchronization process

This chapter explains the manual process for data synchronization.

Once the RDS database is initialized, the tables are defined in the database, but there is no data in the tables. The RDS database needs to be populated with the data from ServiceCenter database. Connect-It scenarios perform the initial database population as well as the periodic data synchronization at a pre-defined time interval.

Connect-It performs the data population or periodic synchronization based on a pre-defined scenario. There are two RDS scenario files: `rds_sc.scn` and `rds_user.scn`. These two scenario files reside in the `/cit` folder of RDS installation directory.

The following describes the general process flow for the manual RDS data synchronization.

Step 1 Start the Connect-It console application.

Step 2 Make sure services for both scenarios, rds_sc.scn and rds_user.scn, exists.

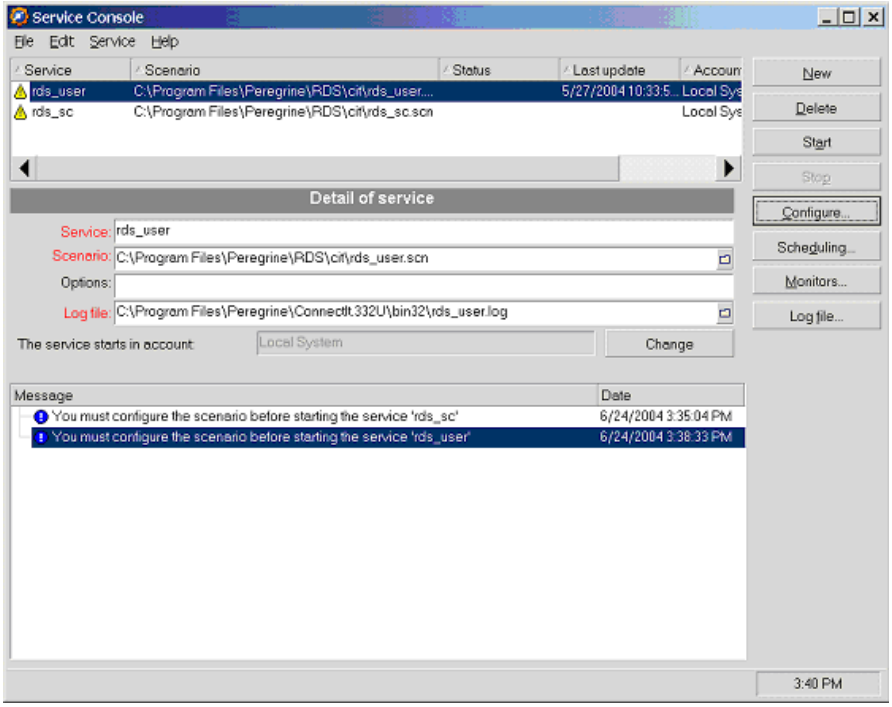
Step 3 Skip this step if the scenarios are properly configured, otherwise; configure both scenarios. This includes configuring the database connection information as well as adjusting time interval for period data synchronization.

Step 4 Once the scenario is properly configured, you can start the services in Connect-It console application. For initial population, delete any pre-existing rds_sc.ini and rds_user.ini files before starting the services. These files reside in the /cit folder in the RDS installation root director. By starting the **rds_sc** service in the console, the service performs the initial data population and then it is scheduled to wake up to perform data synchronization in a pre-defined time interval. The result of this activity is logged in a log file, rds_sc.log. The log file resides in the RDS/logs directory. Starting the **rds_user** service populates the RDS_USER table in the RDS database with the operator information from ServiceCenter, and then it is scheduled to wake up to perform user data synchronization at a predefined time interval.

Step 5 The service will continue to run until you stop the service using the Connect-It console application window.

Using the CIT console to start and stop the RDS scenarios

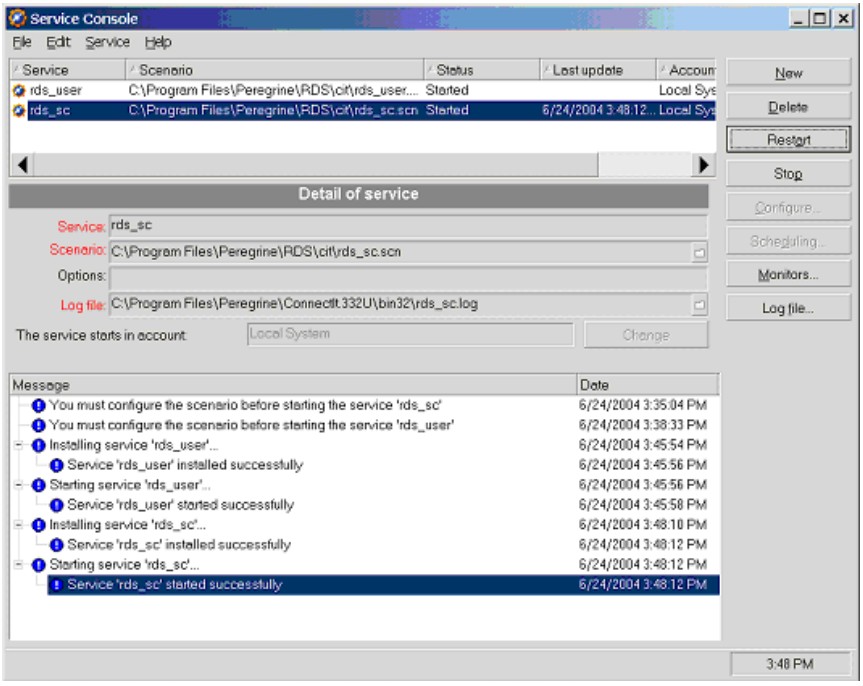
The following figure shows two services, **rds_sc** and **rds_user** in the Connect-It console application.



To start the service

1 Select the service name from the service panel.

2 Click the **Start** button.

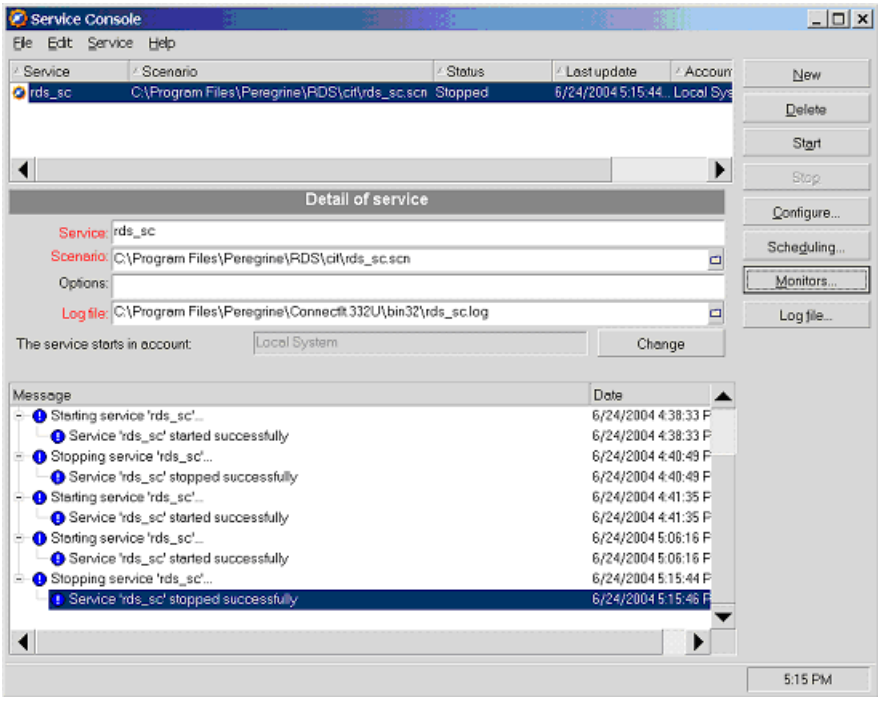


The data synchronization activities are recorded in the log file, rds_sc.log. Information is continuously appended to this file while the service is active. Peregrine recommends that you check the log file to monitor the status of your RDS data synchronization. To verify that the data synchronization did not encounter any errors, search for the string "-53" in the log file. If it is not found, the data synchronization ran without a problem.

To stop the service

1 Select the service name entry from the service list panel.

2 Click the **Stop** button to stop the service.



Using the CIT console to configure the connectors

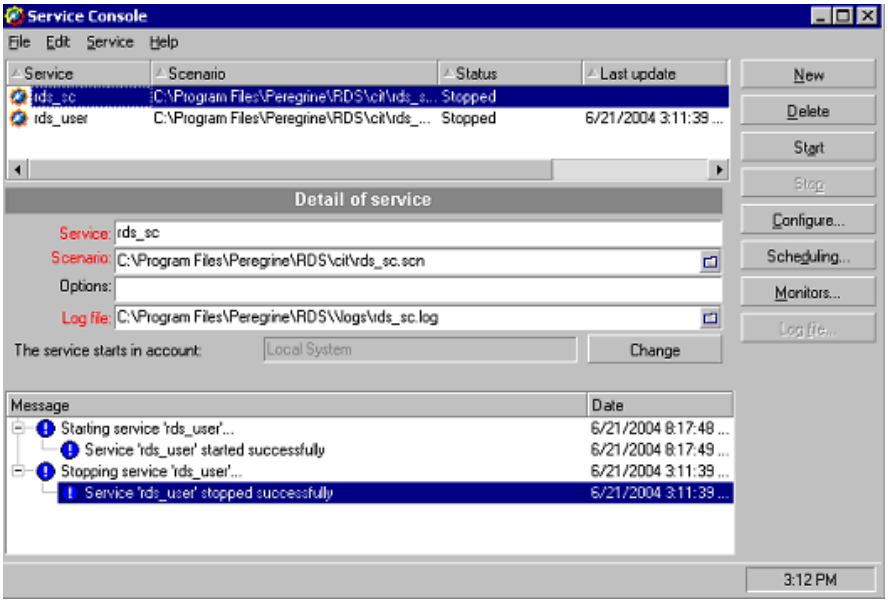
There are two out-of-box Connect-It (CIT) scenarios that come with installation of RDS. These are:

- rds_sc - the scenario, which synchronizes data between ServiceCenter and the RDS database.
- rds_user - the scenario, which synchronizes the operator data between ServiceCenter and the RDS_USER table in the RDS database.

During the installation of RDS, these two scenarios are automatically configured based on the data entered during the RDS installation. If for any reason the RDS installer is unable to automatically configure these scenarios, the system displays an error message to the user. In this case, the problem should be resolved and then the scenarios must be manually configured using the Connect-It Service Console.

To configure the rds_sc scenario

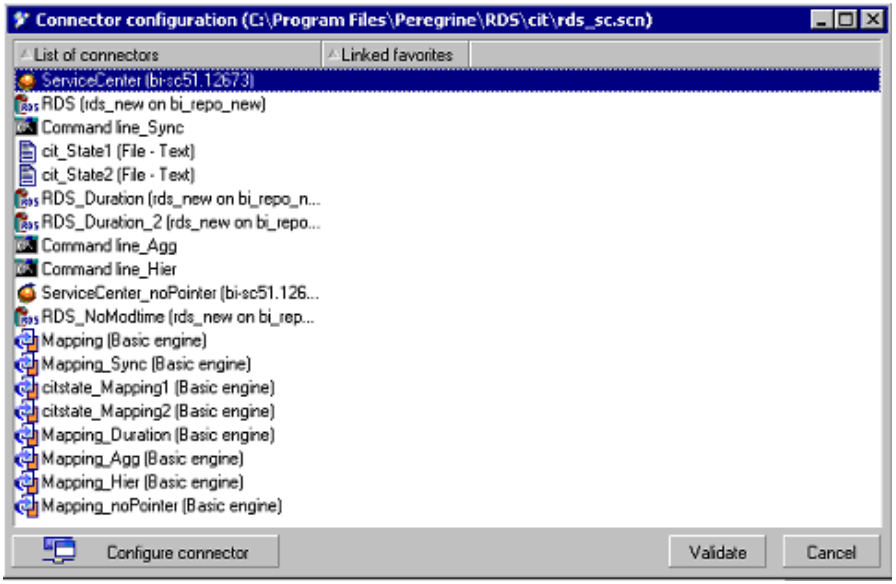
1 Open the Connect-It Service Console.



2 Select the **rds_sc scenario**.

3 Click the **Configure** button.

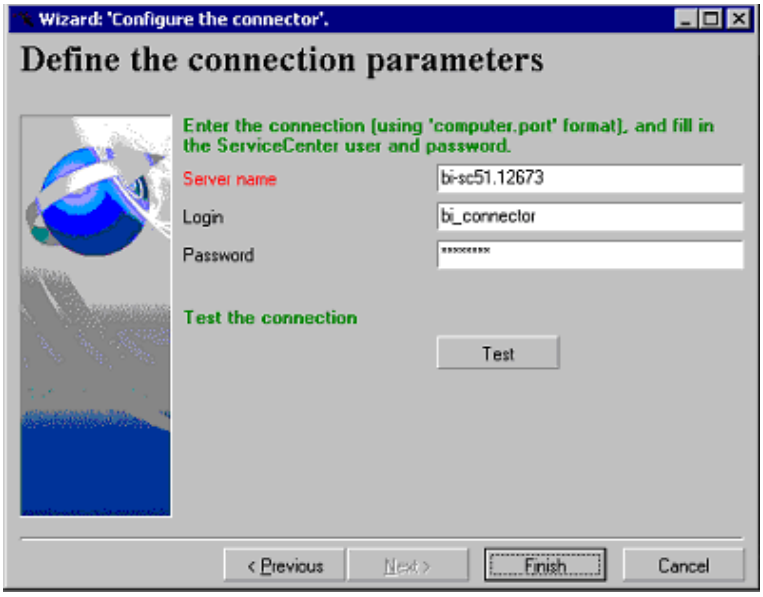
The Connector Configuration window opens and displays a list of connectors.



4 Select each of the following connectors, and click the **Configure** connector button (or simply double click the connector's name).

- ServiceCenter
- RDS
- RDS_Duration
- RDS_Duration_2
- ServiceCenter_noPointer
- RDS_NoModtime

5 Use the ‘Configure the connector’ wizard to enter the correct information. For the ServiceCenter related connectors you will see the following screen.



a Type the ServiceCenter server host name and port number.

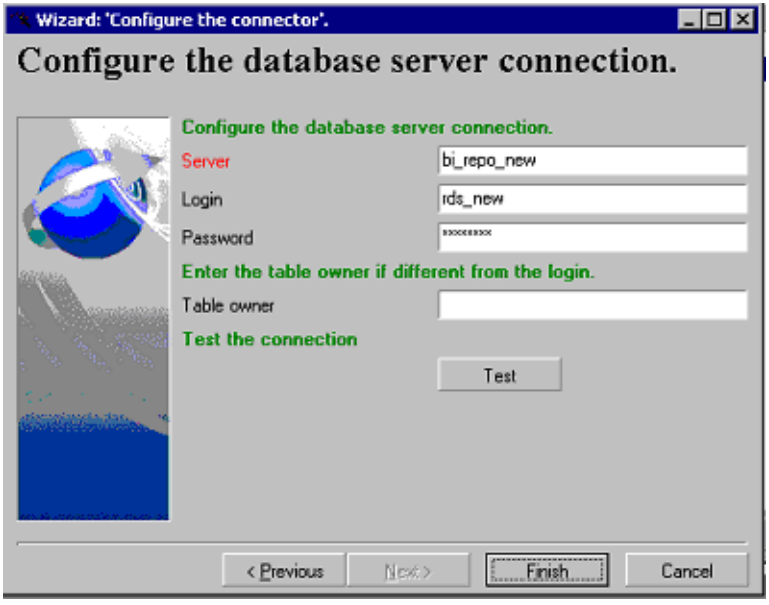
b Type the ServiceCenter default login, which is **bi_connector**.

c Type the Service Center default login password, which is “**passw0rd**”.

d Click **Test** to verify that the connection is ok.

e If the connection is ok, click **Finish**.

6 For the RDS related connectors, you will see the following screen.



a Type the database alias name in Server field.

b Type the user name and password for the RDS database.

c Click **Test** to verify that the connection is ok.

d If the connection is ok, click **Finish**.

7 After you have successfully configured each of the connectors, click the **Validate** button to close the Connector configuration window.

8 Select the **rds_user** scenario.

9 Click the **Configure** button

10 Select the **ServiceCenter** connector, and click the **Configure** connector button (or simply double click the connector's name).

11 Repeat step 5.

12 Select the **RDS** connector, and click the **Configure** connector button (or simply double click the connector's name).

13 Repeat steps 6 and 7.

CHAPTER 3 Customization Workflow

Customizing the Reporting Data Store (RDS) is a complex multi-step process. It is important that these changes are properly planned and executed in the correct order. This chapter, discusses the recommended workflow for creating useful and effective customizations of the RDS; however, this chapter only discusses RDS customization for the simplest case. Anyone planning on any complex customization should first consult with Peregrine Professional Services for assistance.

Note: You should always do any customization in a test environment before implementing the changes in your production environment. You should **not** customize the **rds_user** scenario.

Workflow overview

Step 1 **Planning and Information Gathering** - There are a number of key decisions to be made during the customization process, and these considerations are best done before you even start your modifications. You will need to consider the impact of your changes, and how new/changed/deleted fields and tables will relate to other database objects. Always consider the big picture.

Step 2 **Stopping Programs And Processes** - You need to shut down user access to some programs during the customization process. Be sure to plan for a time when the system gets little traffic, and warn users of impending downtime.

Step 3 **Backing Up Crucial Files** - It's always wise to create copies of any key files you will be editing on the off chance that you will need to back out your changes.

Step 4 **Editing the RDS Schema** - Before you can bring data in from ServiceCenter, that data needs a place to reside. The RDS needs to have additional tables/fields created to accept new ServiceCenter files, including all appropriate links, indices, and aliases.

Step 5 **Editing the Connect-It Scenario** - Once the RDS is prepared, you will need to map your modified ServiceCenter fields to the newly edited RDS fields.

Step 6 **Editing the Universe** - Now that the data is imported into the RDS, you will need to expose that data to your users. You will need to add the new objects to this layer and rename/reorder them in a manner that's meaningful to the end user.

Step 7 **Editing Security Profiles** - If necessary, you may need to edit user access to the data you've changed.

Step 8 **Restarting Programs And Processes** - Finally, the system can be brought online and is ready for testing.

Overview: Planning and Information Gathering

The planning phase of your customization process is perhaps the most crucial step in your project. It is important to map out your changes before you begin.

Who should be involved?

Customization will impact multiple systems in your business intelligence network. Your ServiceCenter administrator, the DBA for your RDS database, and Report Designers should all be consulted. Any IT personnel responsible for bringing your servers up and down should be notified of your timetables. If this is your first time customizing the system, or this is a larger than normal modification, Peregrine recommends that you contact Peregrine Professional Services for a consultation.

Considerations for Adding Data Structures

In addition to simply knowing what ServiceCenter tables and fields you wish to add to your RDS, you need to consider the following options:

- How does this new data link to existing tables? Will you need to define a join in the RDBMS? If so what type? Will you need to create aliases for any tables so the joins will work correctly?
- How should the new data be indexed in the RDBMS?
- Will you be mapping your tables as a DirectMapping table, or will you be creating Dimension, Fact, and Aggregation tables? The latter is more versatile, but also requires more effort to create.
- Will you need to create a Hierarchical table to properly display some of the data?
- Will you be tracking historical data in the new structures? If so, what fields will trigger the creation of a historical record?
- Will new Filter objects need to be created in the Universe?
- Will your field and data level security need to be modified to include the new data? What types of restrictions should be put in place?
- Do Delete Triggers need to be added in ServiceCenter?

Considerations for Modifying and Deleting Existing Data Structures

Important: Deletions are strongly discouraged in the RDS. If a field has been deleted in ServiceCenter, or you no longer wish to expose a field to the end user, it is recommended that you simply remove the mapping from the Connect-It scenario and/or hide the related objects in the Universe.

In addition to the questions above, Modifications and deletions should consider the following:

- Are the data objects being changed linked to other data objects? Is the field being changed used as a link between two tables, as part of an index or unique key combination, or as a member or a parent-child relationship.
- Will deleting or hiding this object cause other data to be orphaned or rendered obsolete?
- If this is not a DirectMapping table, what is the impact this change will have on related Fact and Aggregation tables?

- Are any of the fields being changed flagged for Historical tracking? Will this change require a modification of that flag?
- Will your field and data level security need to be modified to remove non-existing fields or update filter queries?
- Will any of your existing reports have to be altered to reflect the database changes?
- Will Delete Triggers in ServiceCenter need to be removed or modified?

Since all customization is, by definition, unique, this is by no means an exhaustive list of considerations. It is merely a primer for your planning sessions.

Overview: Stopping programs and processes

At a time when your system gets little use, bring down the following services:

- Connect-It Service Console (stop the **rds_sc** scenario and the rds_user scenario).
- Business Objects (BO) server
- Tomcat server

This will allow you to proceed with changes without clashing with existing processes.

Overview: Backing up crucial files

Before making low-level changes to your business intelligence system, it is HIGHLY recommended that the following files be backed up. In the event you are unable to complete the changes successfully, these files will allow you to restore much of your original configuration quickly and accurately.

The following files should be copied:

Note: The names listed here reflect the out-of-box file names. Custom files may have different names.

- RDS_etl.xml - The RDS schema file.
- rds_sc.scn - The Connect-It scenario file.
- rds.unv - The Business Objects Universe file.

- rdms_keywords.xml - Your database reserved words list.
- rds_user.scn - Your Connect-It scenario file.

Overview: Editing the RDS schema

To edit the RDS schema file, you need to start with the XML file that defines the schema: **RDS_etl.xml**. This can be found in the **\conf** folder of the RDS installation root directory.

Navigate to that file and open it in a text editor. Take some time to study the file and its structures. There are many examples of different types of mappings.

Here you will:

- Add new tables and fields.
- Modify or delete existing data.
- Set unique keys.
- Define Dimension, Fact, or Direct Mapping tables.
- Create Aggregated fields.
- Define the fields that trigger Historical records.

When you've made the necessary changes, run the RDS initialization tool to verify the validity of your modifications. If new tables are added and record deletions for the new tables need to be synchronized during the RDS detect sync cycle, you will need to login to ServiceCenter as an administrator and edit your Delete Triggers at this time. See *Chapter 4, RDS Customization* for more details on this procedure.

Overview: Editing the Connect-It scenario

Once the RDS is set to receive the new data, you need to edit the mapping that pulls data from ServiceCenter to the RDS.

Open the file, **rds_sc.scn**, in the Connect-It Scenario Builder. This can be found in the **cit** folder of the RDS installation root directory.

Refresh your view of the database and add, modify, or remove data structures to reflect your changes in ServiceCenter. Then validate and test the scenario.

See *Chapter 5, Connect-It Scenario Customization* for more details on this procedure.

Overview: Editing the universe

Now that the actual data has made its way from ServiceCenter into the RDS, you need to modify the metadata layer to expose this data to your end user.

Login to your Business Objects Designer tool as a user with Designer rights. Navigate to the file rds.unv, which can be found in the directory under the Business Objects installation root directory:

```
\nodes\<server>\<cluster>\universes\<domain>\
```

Here you will:

- Modify the objects to reflect your database changes.
- Define links and aliases for tables.
- Rename and re-order fields into meaningful objects.
- Supply descriptions for universe objects.

Create your changes, then do an integrity check to verify the validity of your links. Once successful, export the universe so it can be viewed online. See *Chapter 6, Business Objects Universe Customization* for more details on this procedure.

Overview: Editing security profiles

This is an optional step, only for those who have set up row level security.

Login to Business Objects Supervisor as a user with Supervisor rights. Edit the rights of appropriate users to the newly modified data structures. See *Chapter 6, Business Objects Universe Customization* for more details on this procedure.

Overview: Restarting programs and processes

If all else has gone smoothly, you are ready to start the system and begin testing. As a final step, restart the following processes in the specified order:

- Business Objects server (WebIntelligence cluster service)
- Connect-It Service Console (start the **rds_sc** scenario)
- Tomcat server

Make sure to test not only that the data appears according to your modifications, but also that aggregations roll up correctly, multiple-table queries create proper joins, security settings filter data correctly, and existing reports all still run correctly.

4 RDS Customization

CHAPTER

This chapter discusses the following topics:

RDS schema customization

You can modify the rds_etl.xml file, which is the out-of-box xml file. In order to modify this file, you will need to have an XML editor tool such as XML Spy. It is important to have a tool that will validate any changes you make to this file against the schema file.

The Reporting Data Store (RDS) Schema requires customization when the ServiceCenter (SC) database on which RDS is deployed has been customized. This section covers three different customization examples:

- New tables are added to ServiceCenter database
- New fields are added to existing ServiceCenter tables
- Existing field attributes are changed

For each of the cases, this section explains how to customize the RDS schema to incorporate the changes in the RDS database. The general process flow for RDS schema customization is:

Step 1 Backup your RDS schema file, RDS_etl.xml.

Step 2 Use an xml editor or a text editor to make the necessary changes to the schema file, RDS_etl.xml. Verify that the changes are valid XML and save the changes.

Step 3 Run the RDS initialization tool to create the RDS database.

Step 4 Verify that the RDS database contains the new table or field.

When you customize the RDS schema, you should perform the following tasks to ensure that any changes you make produce the desired results before you begin any customization.

- Determine what fields and tables you need for the reports you plan to create.
- Follow naming conventions. Peregrine recommends using a prefix (for example, the initials of your company) as part of the name of any new tables or fields you create.

The customization activities described in this section assume that they are for the initial setup, which means the RDS database does not exist yet. Contact Peregrine Customer Support for further assistance if you need to customize an existing RDS database.

Warning: Running the bat file, rds_init.bat against an existing RDS database will result in data loss. The RDS initialization process drops all existing tables and recreates the database based on the currently defined schema.

Adding a new table

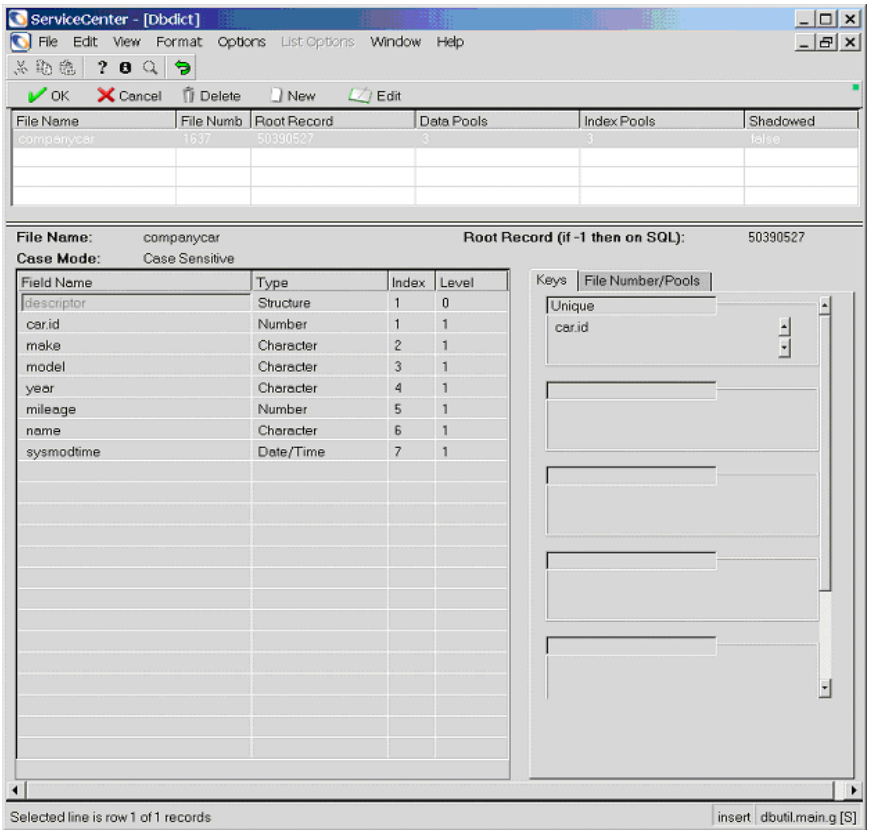
When you add a new table in ServiceCenter and you want to create reports from this new table, you need to use the following procedures to add this new table to the RDS database.

Note: For ease of description, this section uses an example to describe the exact steps you need to do when adding a new table. These steps assume that the new table, *companycar*, has already been added to ServiceCenter database. This table, *companycar,* has seven fields with *car.id* as the primary index.

Peregrine recommends that you always add a system-required field, *sysmodtime* to your new table. This will smooth the RDS data synchronization process through Connect-It.

The following figure shows the table as it has been defined in SC.

Note: The figures in this section are for illustration only. The actual screens that display on your system depend upon which client version of ServiceCenter you are using.



After you add a new table to ServiceCenter, you need to add some data to the table. It is important to note that *sysmodtime* should have some valid date/time data.

To add the companycar table to the RDS database

1 Edit the RDS schema file to incorporate the new table definition.

- First you need to identify which type of table you are adding to RDS. For our example, **companycar** is a directMapping table, so in this example you need to add an instance of a directMapping table into the schema file, as the following:

```
<directMapping name="companycar" DataSourceTableName="companycar">
    <directMappingFields>
      <directMappingField name="name" type="char" size="60"/>
      <directMappingField name="car_id" type="float"/>
      <directMappingField name="make" type="char" size="60"/>
      <directMappingField name="model" type="char" size="60"/>
      <directMappingField name="year" type="char" size="8"/>
      <directMappingField name="mileage" type="long"/>
      <directMappingField name="sysmodtime" type="date"/>
    </directMappingFields>
    <directMappingIndexes>
      <directMappingIndex name="UNIQCOMPANYCAR_IDX">
       <DirectMappingIndexKey fieldName="car_id" srcFieldName="car_id" srctype="float" seqIndex="1"/>
      </directMappingIndex>
    </directMappingIndexes>
</directMapping>
```

a. Open the RDS schema file, **rds_etl.xml**, in an editor and add the above changes to the appropriate section of the schema file.

b. Save the schema file changes you made.

2. Run the RDS initialization batch tool to create the empty table in the database.

   From a dos command window, change the directory to the RDS deployment root directory, then use the following commands:

   a. **C: >cd common/bin**

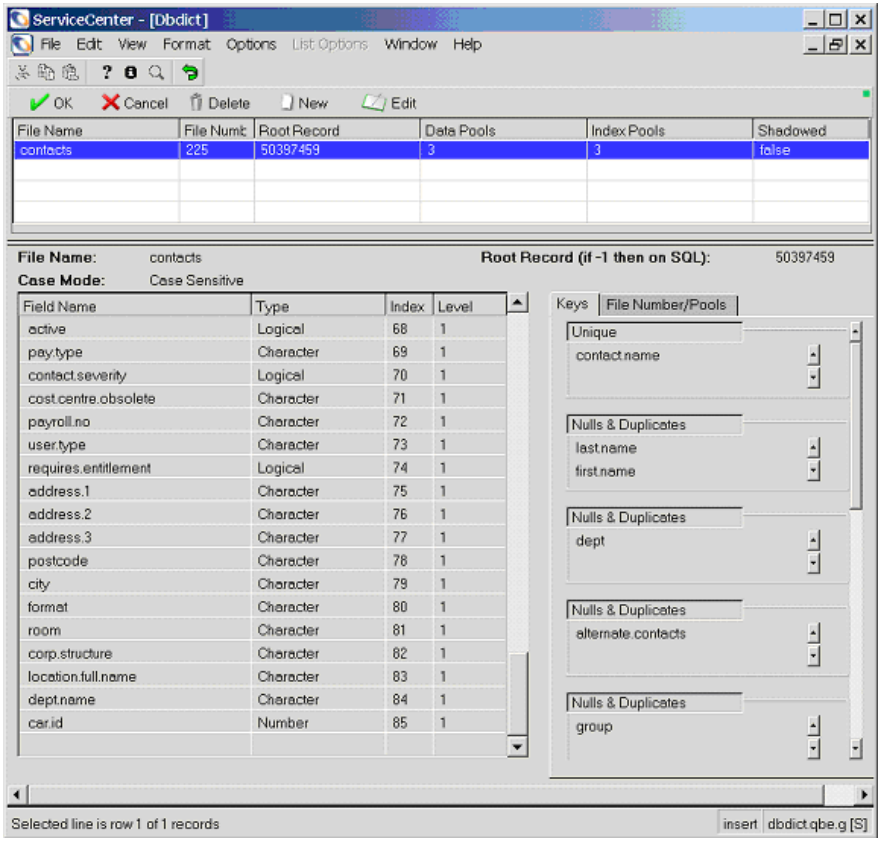
   b. **C: >rds_init**

   When the RDS initialization batch tool finishes initializing the RDS database, the new table should be created.

3. Verify the existence of the new table, with the DBMS vendor's tool; for example, Oracle DBA tool, to ensure that the table was created.

Adding a new field

This examples begins with having a new field, *car.id*, added to an existing ServiceCenter table, *contacts*.



Follow the naming convention listed in *Chapter 1, RDS naming conventions* for naming the new RDS field name. For this example, the new field name in ServiceCenter *contacts* table is **car.id**. Its counterpart RDS field name should be, **CAR_ID**.

To add a new field

1 Add the field definition for **car.id**, as car_id, in the schema file, **rds_etl.xml**, as shown.

```
<dimension name="CONTACT">
  <dimensionTableName>
   CONTACT_D
  </dimensionTableName>
  <dataSourceTableName>
   contacts
  </dataSourceTableName>
  <dimensionTableFields>
   <dimensionTableField name="contact_name" type="char" size="140"/>
   ...(a lot of dimensionTableField definitions are omitted)
   <dimensionTableField name="comments" type="long"/>
   <dimensionTableField name="car_id" type="float"/>
  </dimensionTableFields>
  <uniqueKeys name="contact_unique">
   ...
  </uniqueKeys>
  <facts/>
  <scdKeys/>
  <aggregateKeys/>
</dimension>
```

2 Run the RDS initialization batch tool to add car_id to the existing table, CONTACT_D, in the database.

From a dos command window, change the directory to the RDS deployment root directory, then use the following commands:

a **C: >cd common/bin**

b **C: >rds_init**

When the tool finishes initializing RDS database, the new field should have been added.

3 To verify the existence of the new field in the *CONTACT_D* table, use the DBMS vendor's tool; for example Oracle DBA tool, to ensure that the field exists.

Changing an existing field

When you modify the attributes of an existing field of an existing ServiceCenter table, it might be necessary to make the similar modifications to RDS schema, so the definitions are synchronized. When you rename a data field, it is optional to modify the RDS schema. When you change the data type of a field or change the size of a data field, Peregrine strongly recommends that you make the necessary changes in the RDS schema.

The following example demonstrated how you modify the name of an existing data field. In this example, a ServiceCenter table, *contacts*, has a field, **cost.center**, which has been renamed to **mycost.center**. This example shows how to make the field change in the RDS.

To change an existing field

1 Modify the RDS schema file, by renaming **cost_center** to **my_cost_center** as shown.

```
<dimension name="CONTACT">
   <dimensionTableName>
    CONTACT_D
   </dimensionTableName>
   <dataSourceTableName>
    contacts
   </dataSourceTableName>
   <dimensionTableFields>
    <dimensionTableField name="contact_name" type="char" size="140"/>
    ...
    <dimensionTableField name="my_cost_center" type="char" size="30"/>
    ...
    <dimensionTableField name="comments" type="long"/>
    <dimensionTableField name="car_id" type="float"/>
   </dimensionTableFields>
   <uniqueKeys name="contact_unique">
    ...
  </uniqueKeys>
   <facts/>
   <scdKeys/>
   <aggregateKeys/>
</dimension>
```

Note modified field name

2 Run the RDS initialization batch tool to the field renamed from cost_center to my_cost_center in the existing table, **CONTACT_D**, in the database.

   From a dos command window, change the directory to the RDS deployment root directory, then use the following commands:

   a **C: >cd common/bin**

   b **C:>rds_init**

   When the tool finishes initializing RDS database, the name of the existing field should have changed.

3 To verify the changes to the field, use the DBMS vendor's tool; for example Oracle DBA tool, to ensure that the name has been properly updated.

ServiceCenter trigger customization

Since the RDS database shadows the ServiceCenter database, a record deletion taken place inside ServiceCenter also needs to be properly synchronized in the RDS. To accomplish this, the RDS tracks delete events of the various ServiceCenter tables that the RDS cares about. The RDS does this by having ServiceCenter log record deletion events in ServiceCenter.

At the RDS data synchronization time, the RDS tool processes these deletion events. For each of the deletion events in ServiceCenter, the corresponding RDS record is processed and marked accordingly with an inactive status.

Predefined SC triggers

To track deletion events in ServiceCenter, BI Portal uses predefined post-delete triggers for each of the following tables in ServiceCenter.

SC Table Name	RDS Table Name	Table Type
activity	ACTIVITY	DIRECT_MAPPING
assignment	ASSIGNMENT	DIRECT_MAPPING
category	CATEGORY	DIRECT_MAPPING
clocks	CLOCK	DIMENSION
cm3r	CM3R	DIMENSION
cm3rcatphase	CM3RCATPHASE	DIRECT_MAPPING
cm3rgroups	CM3RGROUPS	DIRECT_MAPPING
cm3sla	CM3SLA	DIRECT_MAPPING
cm3t	CM3T	DIMENSION
cm3tcatphase	CM3TCATPHASE	DIRECT_MAPPING
cmlabor	CMLABOR	DIMENSION
cmparts	CMPART	DIMENSION
company	COMPANY	DIMENSION
computer	COMPUTER	DIRECT_MAPPING
contacts	CONTACT	DIMENSION
country	COUNTRY	DIRECT_MAPPING

SC Table Name	RDS Table Name	Table Type
dept	DEPT	DIMENSION
device	DEVICE	DIMENSION
deviceparent	DEVICEPARENT	DIRECT_MAPPING
displaydevice	DISPLAYDEVICE	DIRECT_MAPPING
expline	EXPLINE	DIMENSION
furnishings	FURNISHINGS	DIRECT_MAPPING
handhelds	HANDHELDS	DIRECT_MAPPING
incidents	INCIDENT	DIMENSION
location	LOCATION	DIMENSION
mainframe	MAINFRAME	DIRECT_MAPPING
model	MODEL	DIMENSION
networkcomponents	NETWORKCOMPONENTS	DIRECT_MAPPING
officeelectronics	OFFICEELECTRONICS	DIRECT_MAPPING
operator	OPERATOR	DIMENSION
outage	OUTAGE	DIMENSION
outagedetail	OUTAGEDETAIL	DIRECT_MAPPING
pcsoftware	PCSOFTWARE	DIRECT_MAPPING
probcause	PROBCAUSE	DIRECT_MAPPING
probsummary	PROBSUMM	DIMENSION
resolution	RESOLUTION	DIRECT_MAPPING
rootcause	ROOTCAUS	DIMENSION
screlation	SCRELATION	DIRECT_MAPPING
servicecontract	SERVICEC	DIMENSION
serviceent	SERVICEENT	DIRECT_MAPPING
servicereviews	SERVICEREVIEWS	DIRECT_MAPPING
sla	SLA	DIMENSION
slamonthly	SLAMONTHLY	DIRECT_MAPPING
slamonthlyag	SLAMONTHLYAG	DIRECT_MAPPING
slaresponse	SLARESPONSE	DIRECT_MAPPING

SC Table Name	RDS Table Name	Table Type
software	SOFTWARE	DIRECT_MAPPING
softwarelicense	SOFTWARELICENSE	DIRECT_MAPPING
storage	STORAGE	DIRECT_MAPPING
telecom	TELECOM	DIRECT_MAPPING
vendor	VENDOR	DIMENSION

Add and remove triggers for a user-defined ServiceCenter table

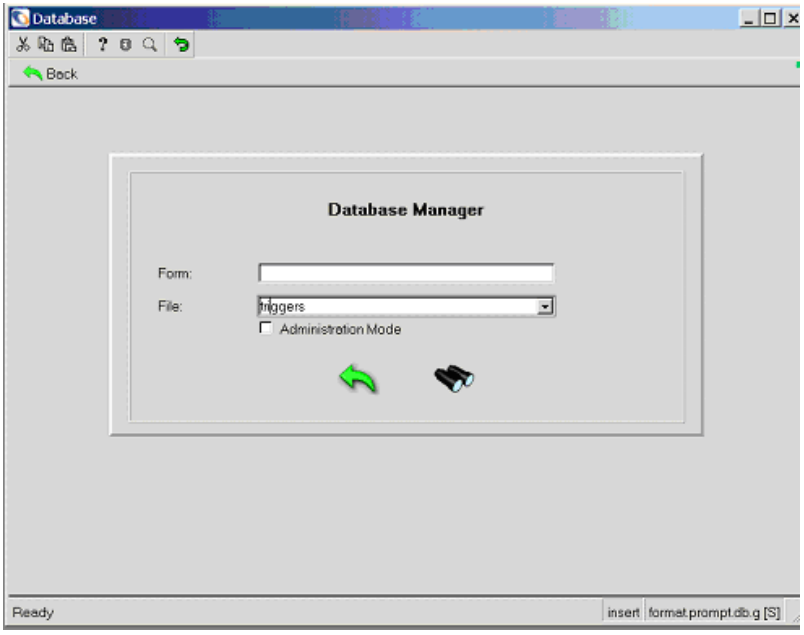
If you add a new ServiceCenter table and this new table requires the RDS record deletion synchronization support, you need to manually add a post-delete trigger for this table in ServiceCenter.

Note: The screen shots in this section are for illustration purposes only. The version of ServiceCenter you are using may have a different user interface.

To define a post-delete trigger for a new ServiceCenter table (for example, if the table name is companycar)

1. Log on to ServiceCenter as bi_admin or some other ServiceCenter user that has administrative rights.
2. From the Main Menu, select **Toolkit** > **Database Manager.**

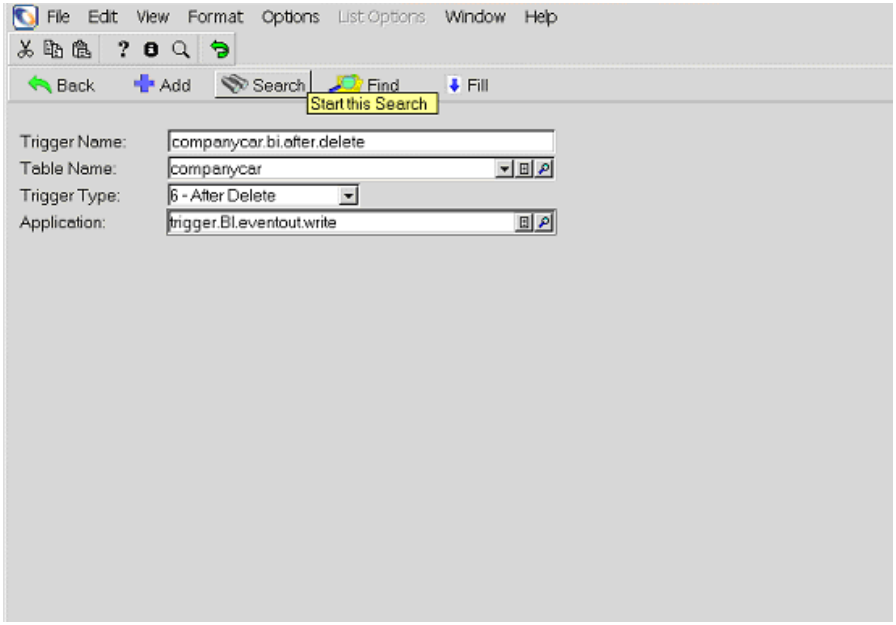
The Database Manager page opens.



3 Type or select the table name, **triggers**, in the text box labeled File.

4 Click **Search**.

The Triggers Search page opens.



5 Type the following information:

Trigger Name	companycar.bi.after.delete
Table Name	companycar
Trigger Type	After Delete
Application	trigger.BI. eventout.write

6 Click **Add** to add this trigger definition to SC.

You can also remove a pre-defined trigger from SC if you find that you no longer need it.

To remove a trigger for a new SC table

1 Log on to SC as bi_admin or some other SC user that has administrative rights.

2 From the Main Menu, select **Toolkit** > **Database Manger**.

The Database Manager page opens.

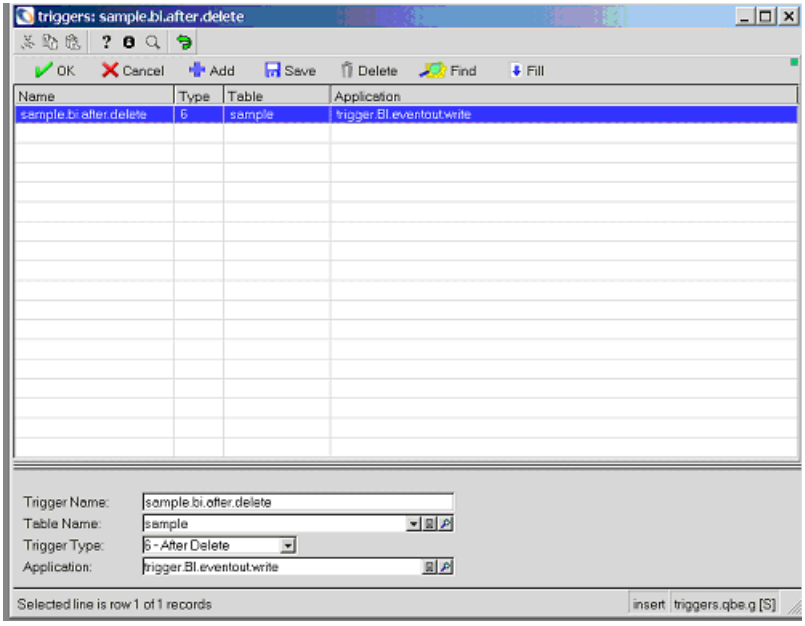
3 Type or select the table name, **triggers**, in the text box labeled File.

4 Click **Search**.

The Triggers Search page opens.

5 Type at least one search parameter; for example, the name of the trigger you want to delete.

The Triggers Search page displays the trigger record you requested or a list of records.



6 If there is a list of records, select the record you want to delete and click **Delete**; otherwise, just click **Delete**.

The system asks you to confirm the delete request.

5 Connect-It Scenario Customization

CHAPTER

This chapter discusses the steps to follow when you are doing BI Portal customization that requires you to modify some of the more complex components of BI Portal. In addition, this chapter discusses how to customize the Reporting Data Store (RDS) Connect-It (CIT) scenario, rds_sc.scn, for ServiceCenter. The RDS uses this scenario to perform data synchronization. This chapter also covers how to add a new data mapping, how to update existing mapping, how to change the connector configuration, and how to change the scenario scheduler. The topics covered in this chapter are:

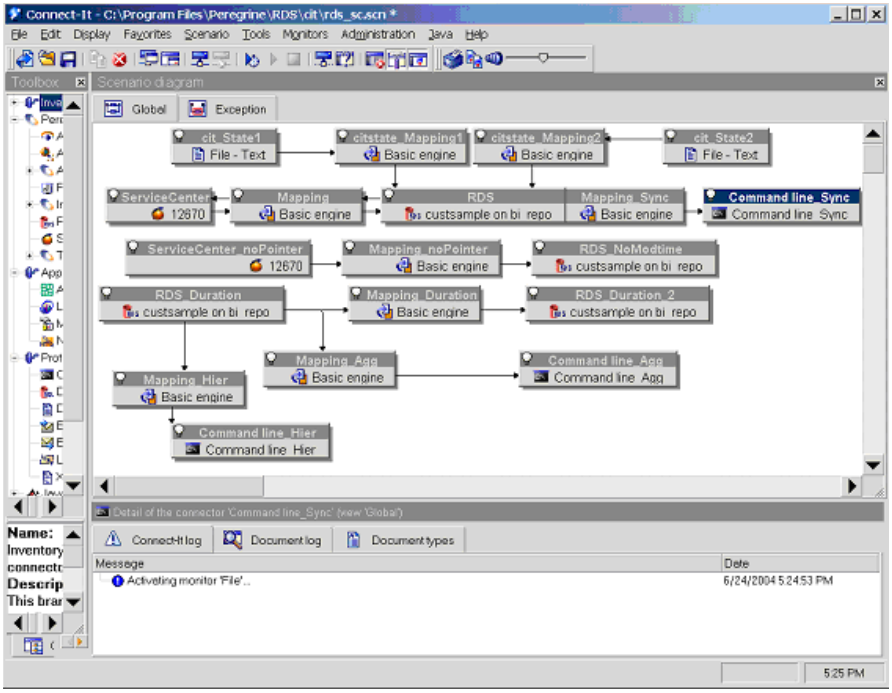
Using CIT Scenario Builder tools to view an RDS scenario

The RDS Connect-It scenario is made up of a collection of connectors. To view the detailed mapping information of the connectors, use the Connect-It Scenario Builder tool.

To view the RDS scenario

1 In the Connect-It Scenario Builder tool from the File menu, double click, **rds_sc.scn**.

The scenario opens in the Scenario diagram window.

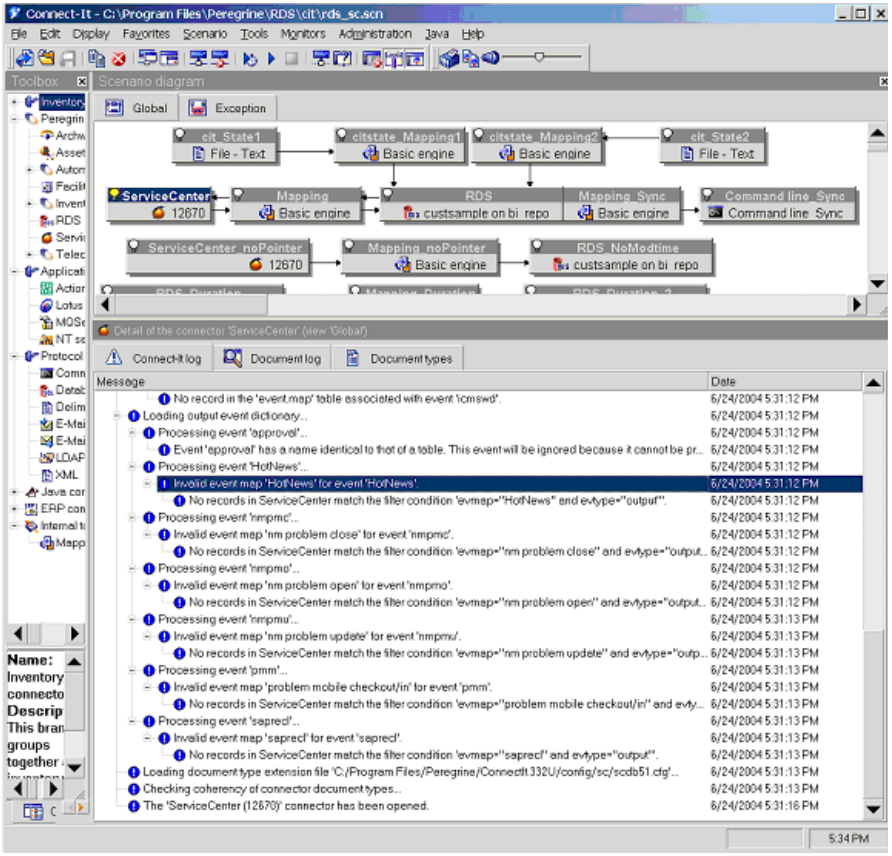


2 Select the connector ServiceCenter.

3 Right click and select **Open** connector in the popup menu.

This essentially instructs Connect-It to open the data source, ServiceCenter, and make all of the document types available to be selected and mapped to the destination database, the RDS.

Connect-It displays a list of status message in the message window highlighting information as it opens available document types defined in ServiceCenter (the source) against those defined in the scenario. If it finds any inconsistency between what is known in the scenario and what is defined in the ServiceCenter, the message is flagged with a red stop sign.



You can use the Connect-It Scenario Builder tool to make the following modifications to the existing scenario:

- Add or remove document types
- Change existing mapping information
- Update scheduler settings

Adding new mapping

You need to add a new data mapping to the existing scenario (rds_sc.scn), if you customized your ServiceCenter database by adding new tables or new data fields to existing tables. To better describe how to add a new data mapping, this section uses the following example to demonstrate how you can add a new table to the existing scenario and how you can add a new field to an existing table in the scenario.

Adding mapping for a new table

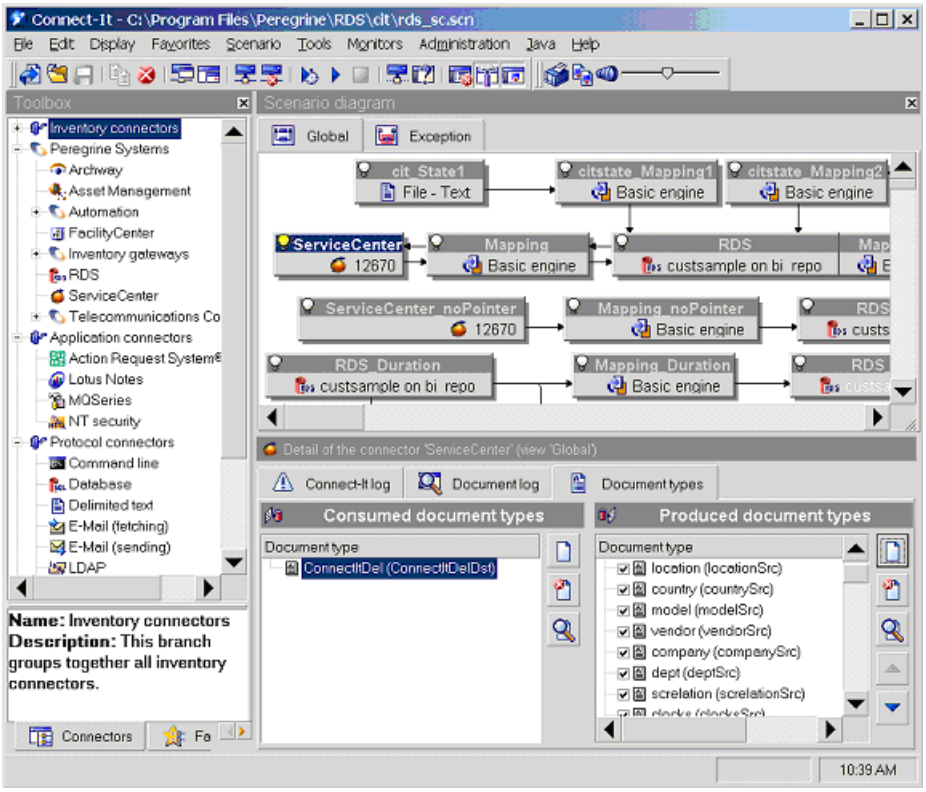
As with the example given in *Chapter 4, RDS Customization* shows, when you add a new table, *companycar*, in ServiceCenter you need to do additional customization to make the data available for reports. The table, **companycar**, is created in the RDS database at RDS initialization time. The table has seven fields, with **car.id** the unique key to the table. This section presents the steps to customize the RDS scenario file, rds_sc.scn, to include the mapping information for the new table so at RDS data synchronization time the data in the ServiceCenter **companycar** table can be populated to the RDS **companycar** table.

To add mapping for a new table

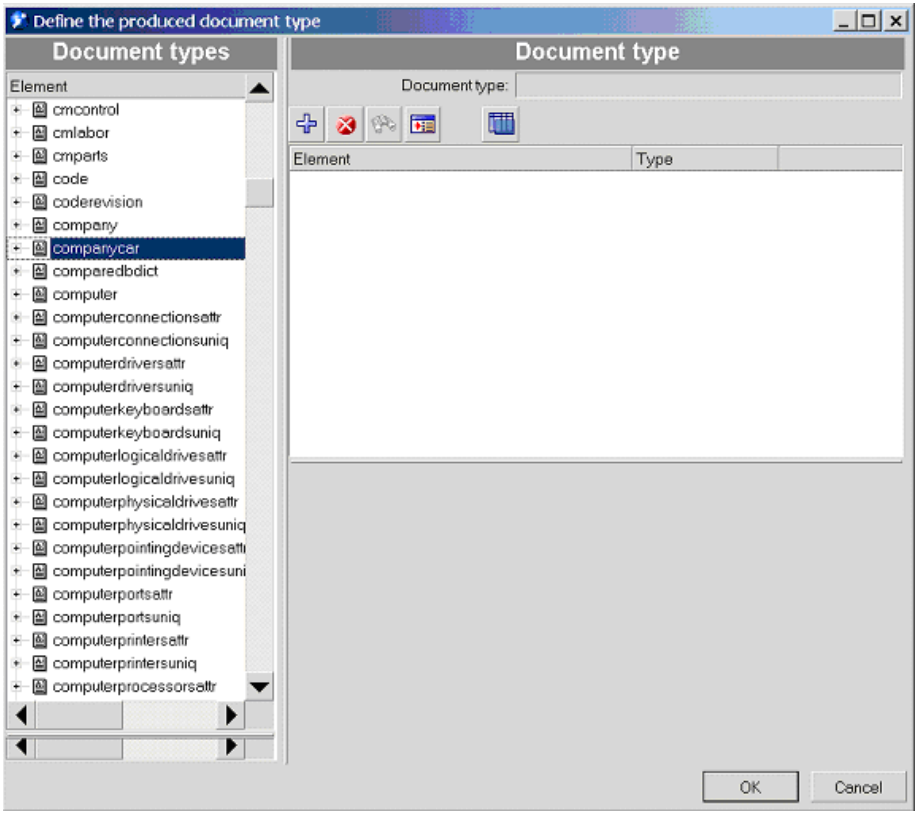
1. Open rds_sc.scn in your Connect-It Scenario Builder tool.
2. Find and select the connector node with label: **ServiceCenter**, then right click to display "open connector" from the pop-up menu.

   The connector opens.
3. Select the Document Types tab.

4 Click the **Create** button under the pane titled, Produced Document Types.

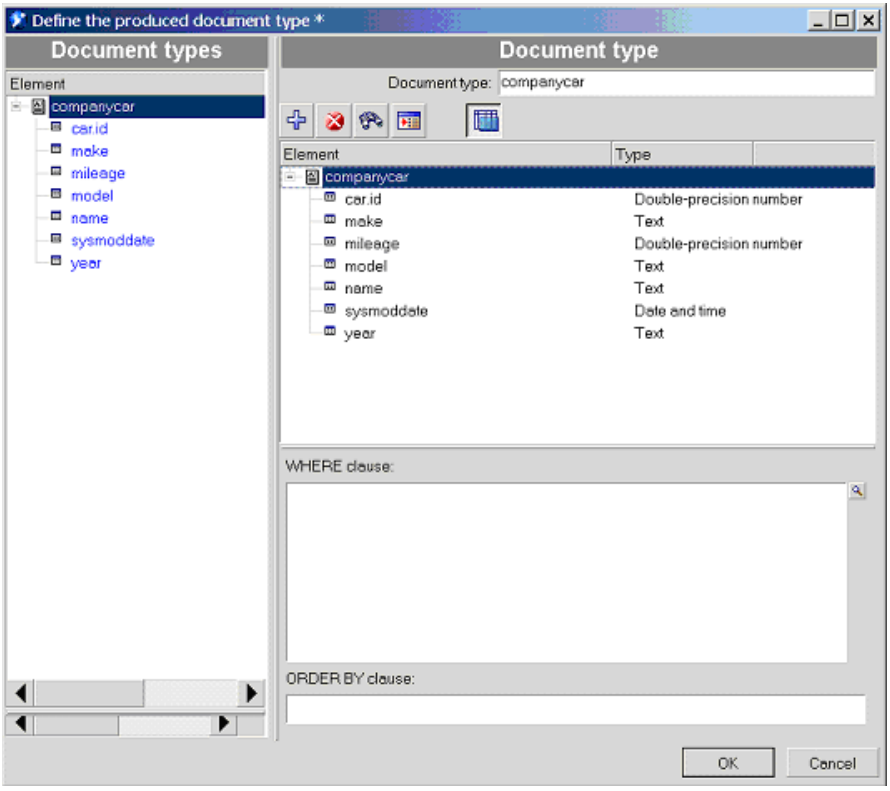


The Define the produced document type window opens.



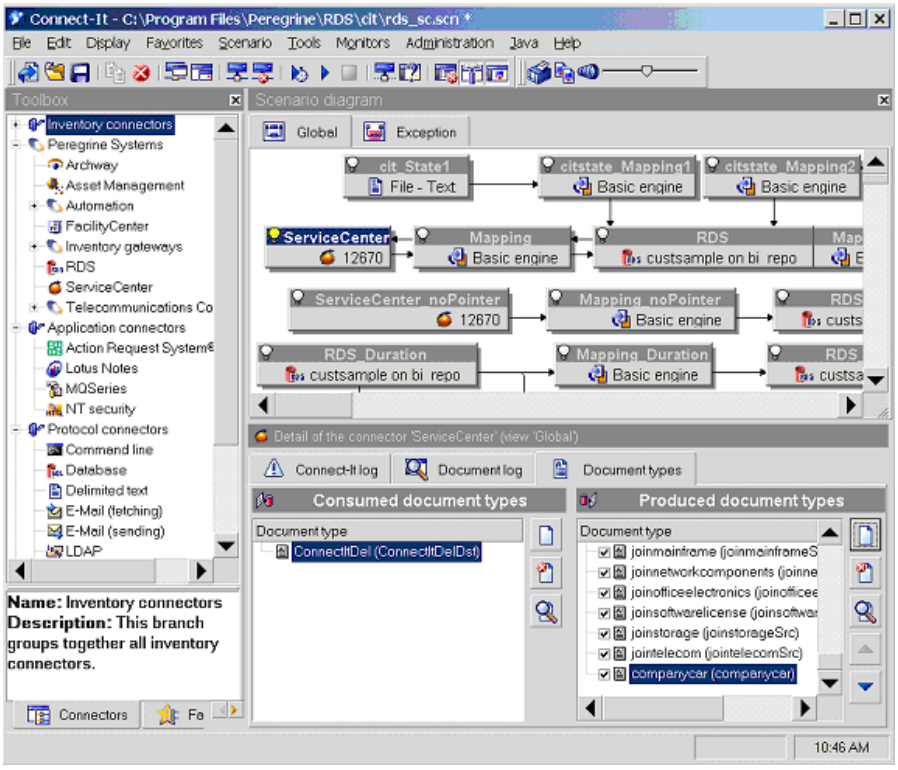
5 In the Define the produced document type window, select the new table, **companycar**, from document types list box.

6 Click the button with "+" on it to expand the table and display the fields.

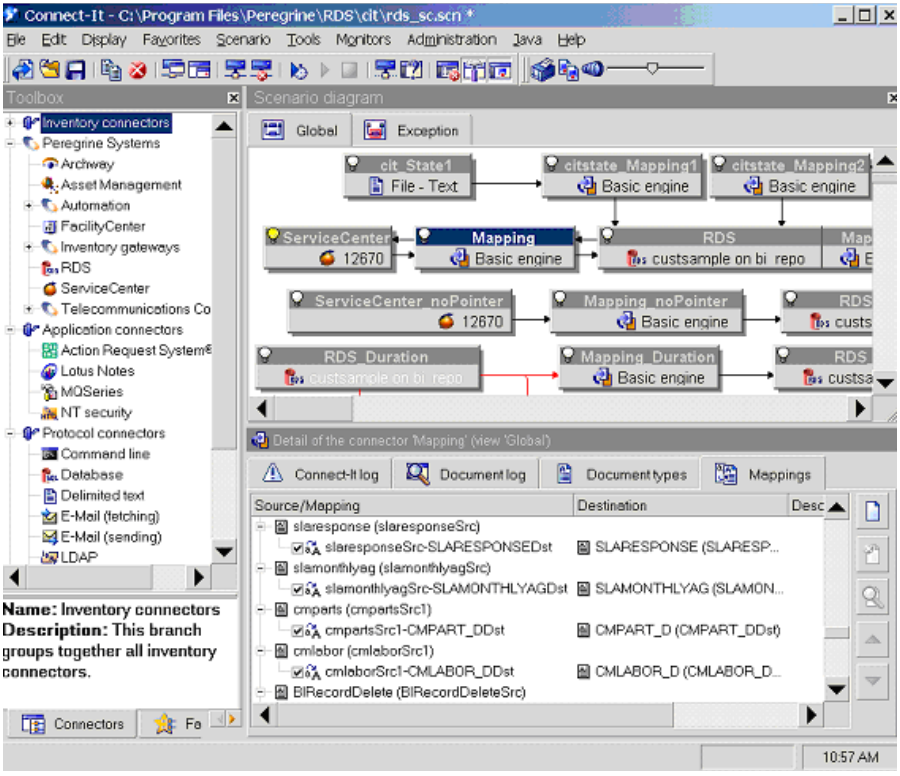


7 Click **OK** to add the new table as a new document type and dismiss this window.

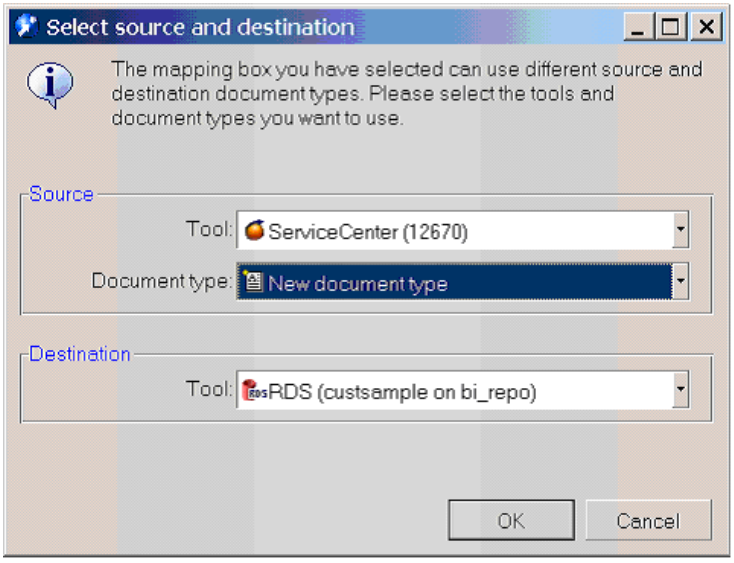
The table, *companycar*, appears as a new document type in the Produced document types pane.



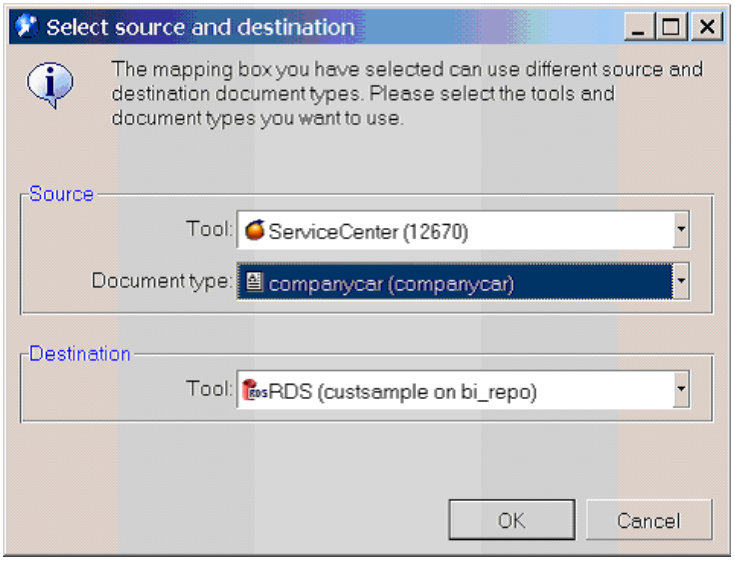
8 Select the connector node, **Mapping,** and click the **Mapping** tab in the lower pane. Click the **Create a New Mapping** button.



The dialog box to add the new mapping opens.

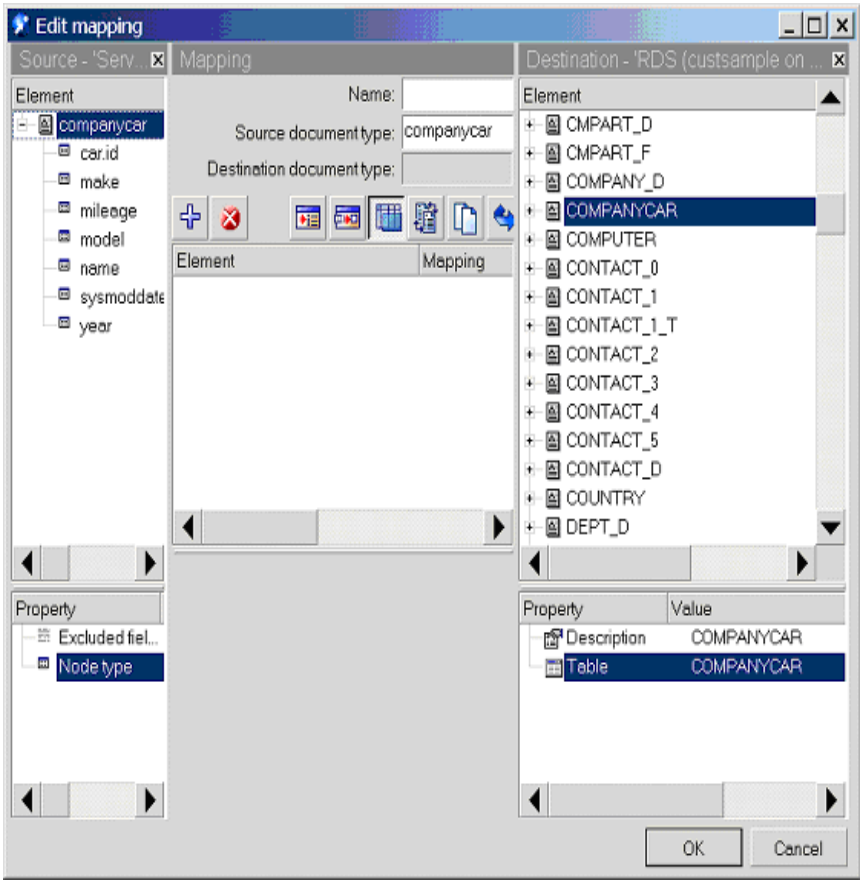


9 Select **companycar** from document type drop-down list.



10 Click **OK** to dismiss the dialog box. Select Yes to save the scenario changes first.

After Connect-It finishes processing the new mapping definition, a new mapping window opens for **companycar**.



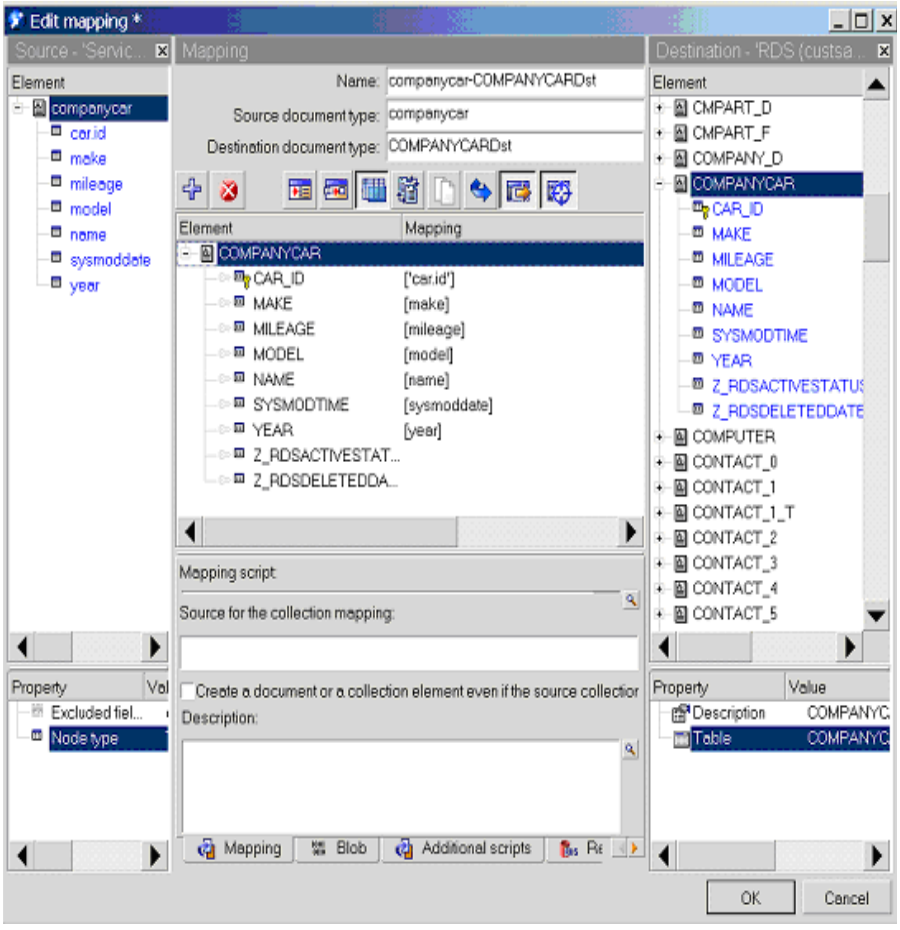
11 To edit the mapping for companycar, type **companycar** in the Name field.

12 Select COMPANYCAR from the destination list box

13 Click add button "+" to add the mapping from source table to destination table.

14 To add the field mapping information from a source table to its destination table, use the mouse to select a field from the source table and then drag the selected field to the corresponding field in the mapping element list of the destination table.

For example, **car.id** of companycar maps to CAR_ID of COMPANYCAR, so selecting **car.id** and dragging the selection to CAR_ID causes the mapping ['car.id'] to be created visually.

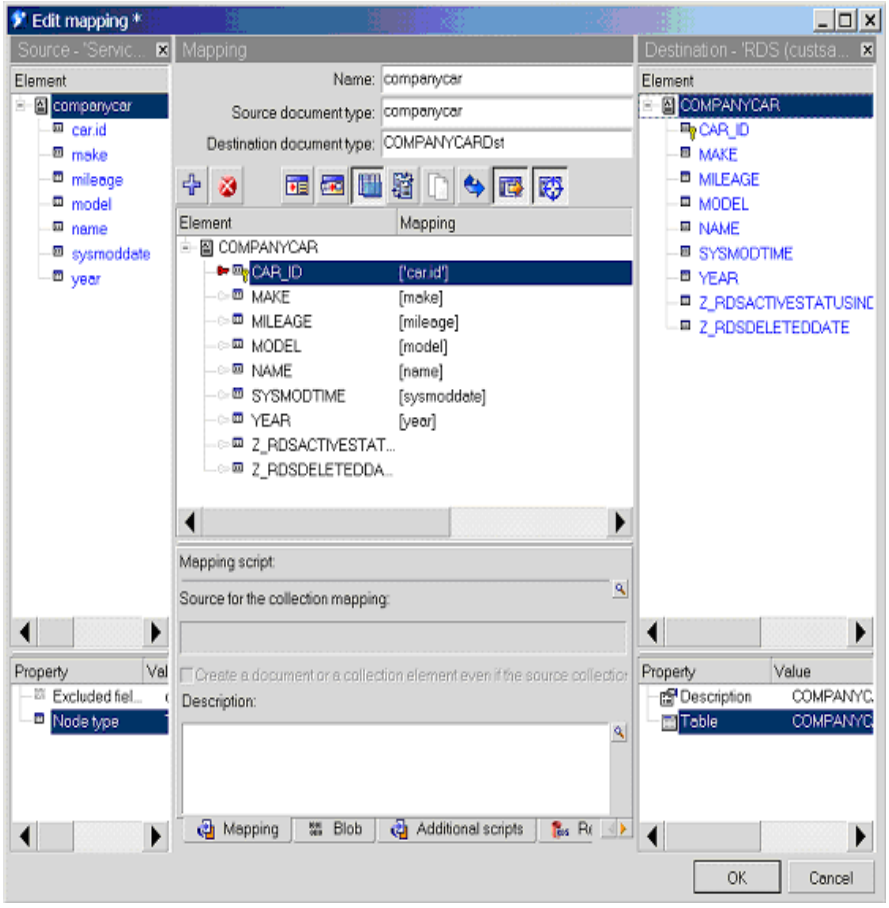


When you finish with adding all field mappings, you now need to define the primary key field.

Define the primary key field of the new table for CIT

1 In this example, the primary key for companycar is **CAR_ID**. In the mapping panel (in the middle pane), click the tiny transparent key icon in front of the field **CAR_ID.**

It turns red.



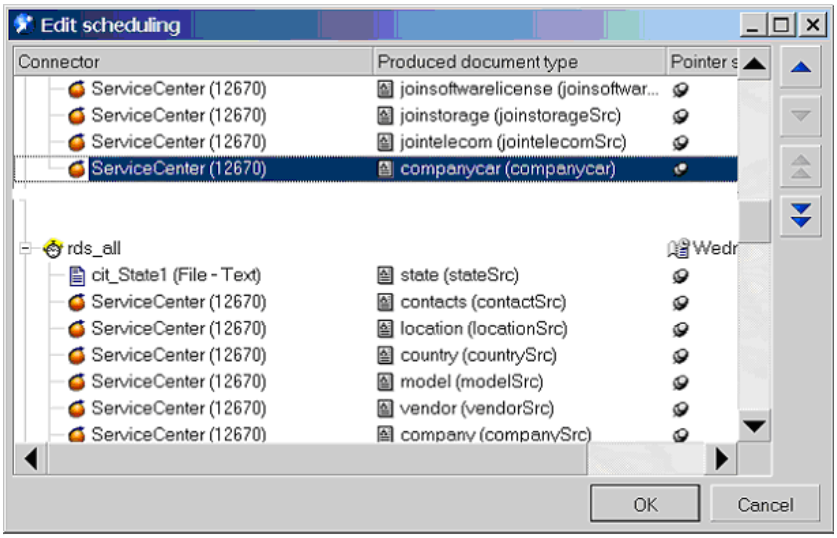
At this point, you have finished defining the primary key.

2 Click **OK** to dismiss the editing mapping session. At this point, you have successfully added a new table mapping to the RDS scenario.

To add the new table to the scheduling process

1 In the Connect-It Scenario Builder window from the top menu, select Scenario Scheduling, to edit the scheduling.

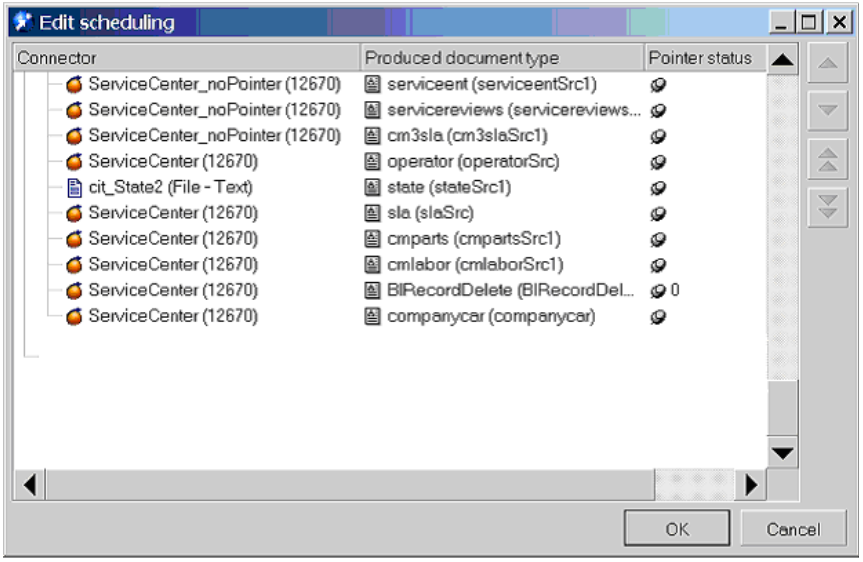
The Edit Scheduling window opens.



2 Scroll down the list of table names for connector “Once”.

3 Select **companycar** under Produced document type column.

4 Drag-and-drop **companycar** entry from connector “Once” to connector “rds_all”.



You have finished the process of adding a mapping for a new table. Now save the updated scenario and use Connect-It console to run the updated scenario. Now you are ready to use the scenario to perform data synchronization.

Adding a new field to an existing mapping

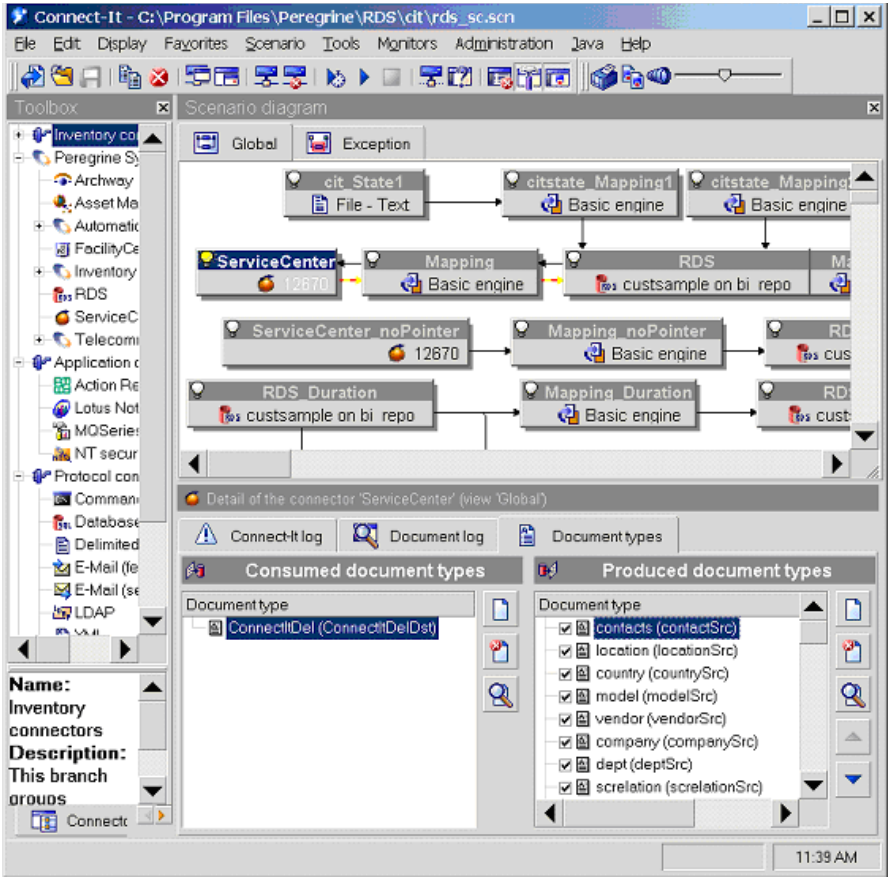
In the previous chapter, you were given steps to add a new field, **car.id**, to the *contacts* table in ServiceCenter and how to add the **car_id** field to the **contact_d** table in the RDS. In order for the Connect-It scenario to populate this new field in the RDS with data, you must modify the RDS scenario to include mapping information for the new field. This example shows how you can add the new field, **car.id**, to an existing table mapping in rds_sc.scn.

To add a new field to rds_sc.scn

1. Open rds_sc.scn in your Connect-It Scenario Builder tool.
2. Find and select the connector node with label: **ServiceCenter**, then right click to display "open connector" from the pop-up menu.

   The connector opens.
3. Add new field **car.id** to the **contactSrc** document type.
   a. Click the Document Type tab.
   b. Select contacts (contactSrc) element in the Produced Document Type pane.

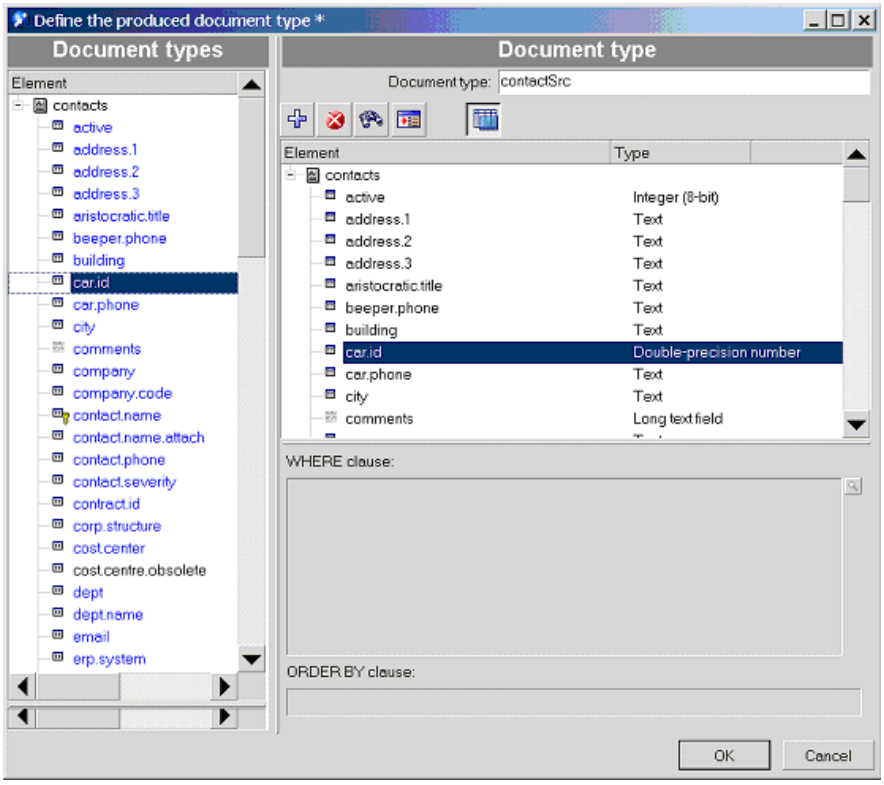
c Click Edit to edit the detail fields of the contacts table.



4 Edit the document type, **contactSrc,** and add the new field **car.id**.

a Select the **car.id** field in the element list box for the contacts table on the left-hand pane of the window and click the Add "+" button to add the new field to **contactSrc**.

b Click OK to dismiss the edit document type window.

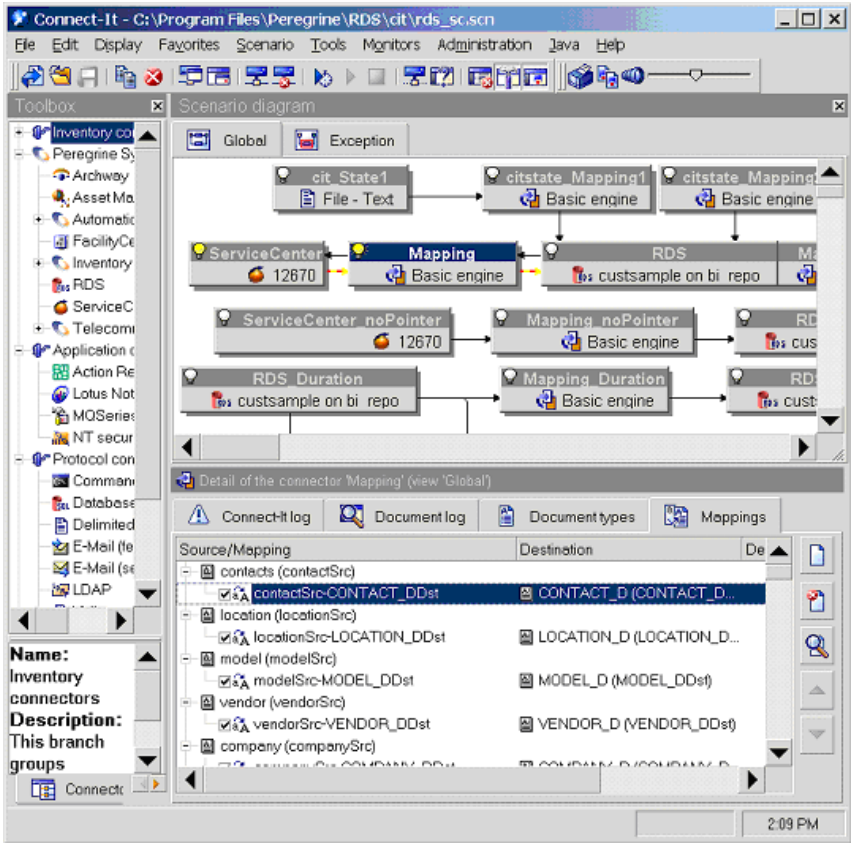


5 Add new field mapping for **car.id**. Select the connector the label: mapping.

a Click the Mappings tab.

b Select mapping relationship of **contactSrc** to CONTACT_D.

c Click **Edit** to open the Edit mapping window for this mapping.

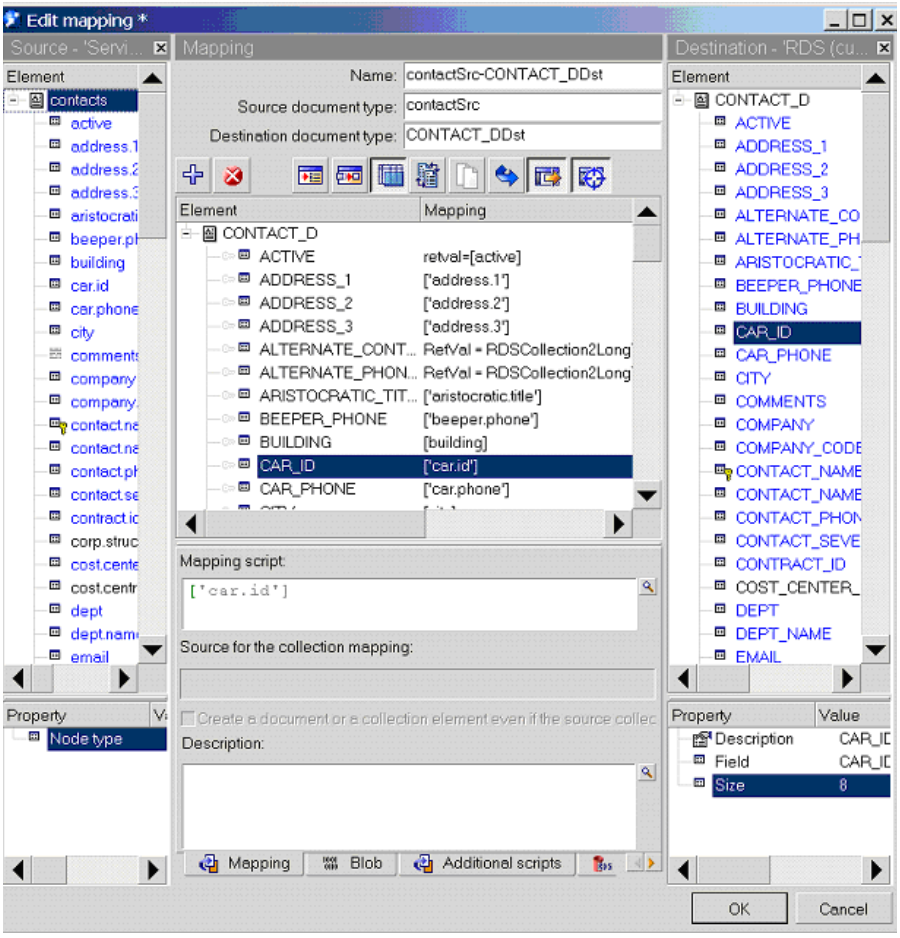


d Select the field **CAR_ID** from the destination element list box

e Click the Add "+" button to add to the mapping element list.

f Select the field **car.id** from source element list box and drag-and-drop it on the CAR_ID element in the Mapping element list box.

g Click OK to finish the mapping definition.



Test your customized RDS scenario

Once you customize the RDS scenario, you need to run the scenario to perform the data synchronization process. Refer to *Chapter 3, Customization Workflow* for more detail.

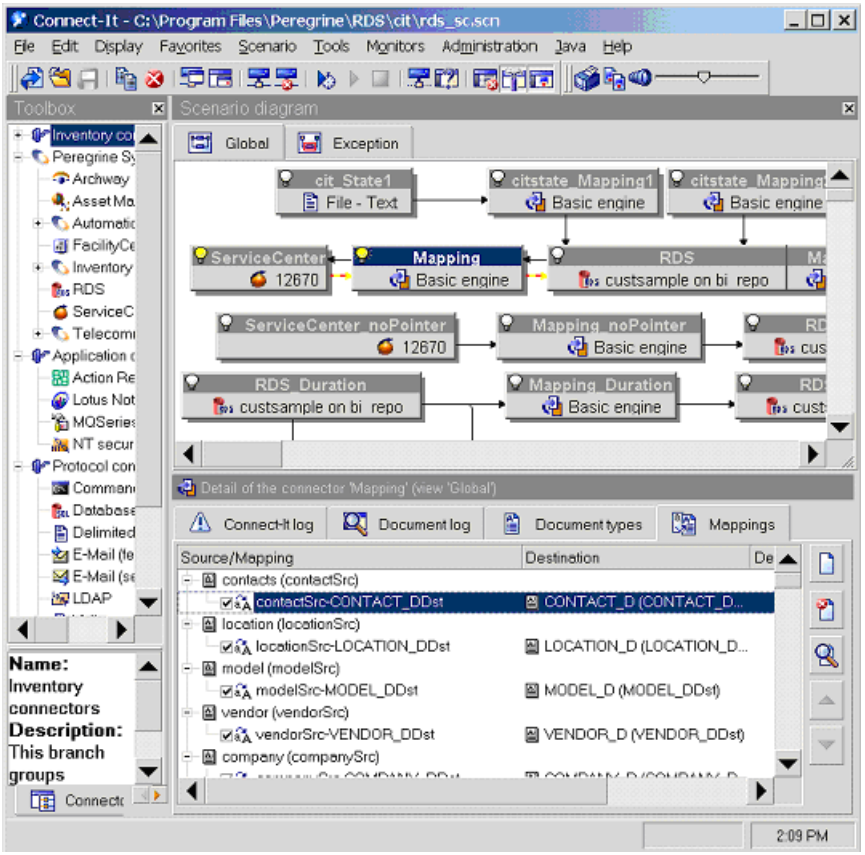
Update existing mapping

If attributes for existing fields in ServiceCenter database as well as the RDS database have been modified, then the mapping information for these changed fields needs to be modified in the RDS rds_sc.scn scenario file.

The following example demonstrates how you can incorporate changes for a field name; for example, *cost.center*.

To update mapping information for a field

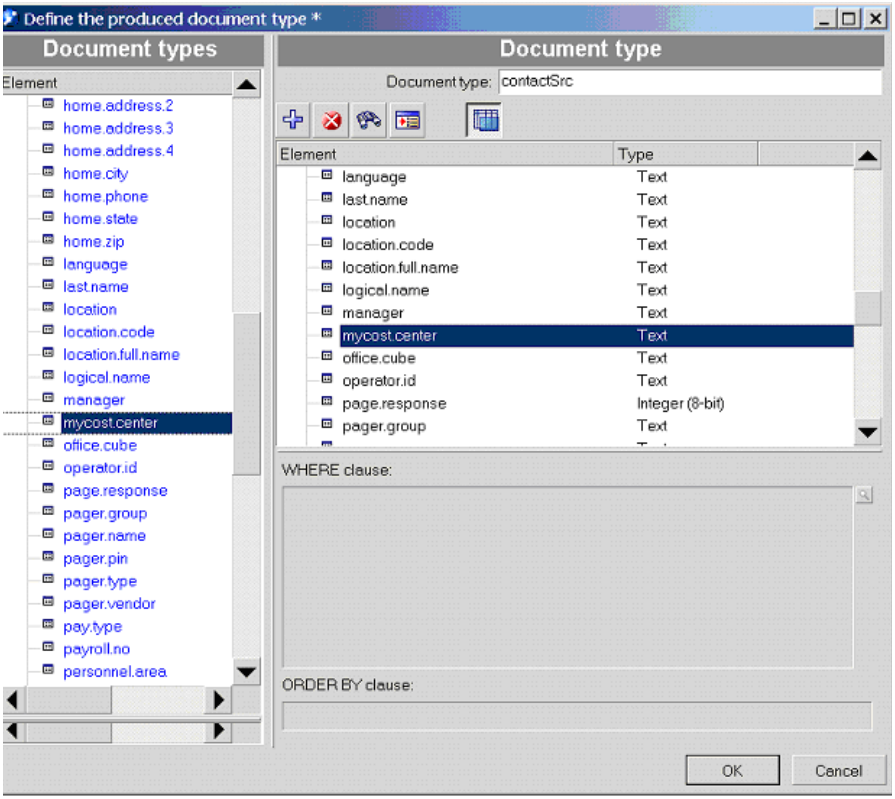
1. Open rds_sc.scn in your Connect-It GUI tool.
2. Find and select the connector node with label: **ServiceCenter**, then right click to display "open connector" from the pop-up menu.
3. Click the Document Type tab.
4. Select contacts (contactSrc) element in the Produced Document Type pane.



5 Select the field, **mycost.center**, from contacts document types list box on the left-hand side of the window.

6 Click the Add "+" button to update the mapping for the renamed field to **contactSrc**.

7 Click the **OK** button to dismiss the window.

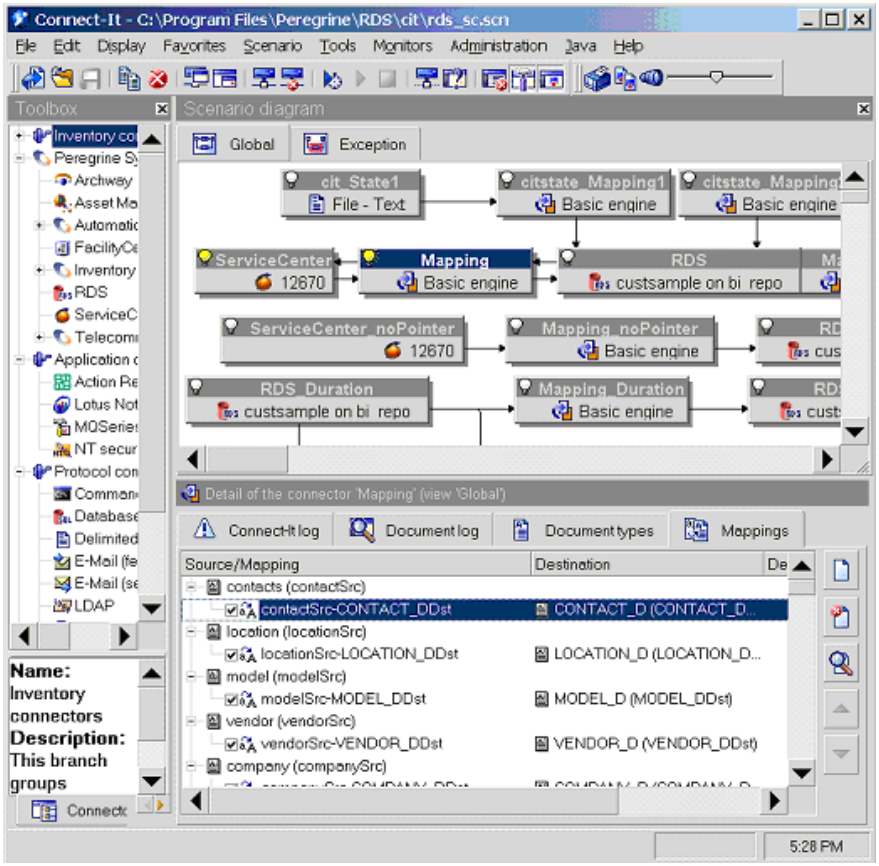


8 Update the mapping information.

a Select the Mapping connector node and right click to open the connector.

b Select **contactSrc-CONTACT_DDst** from Mappings tab.

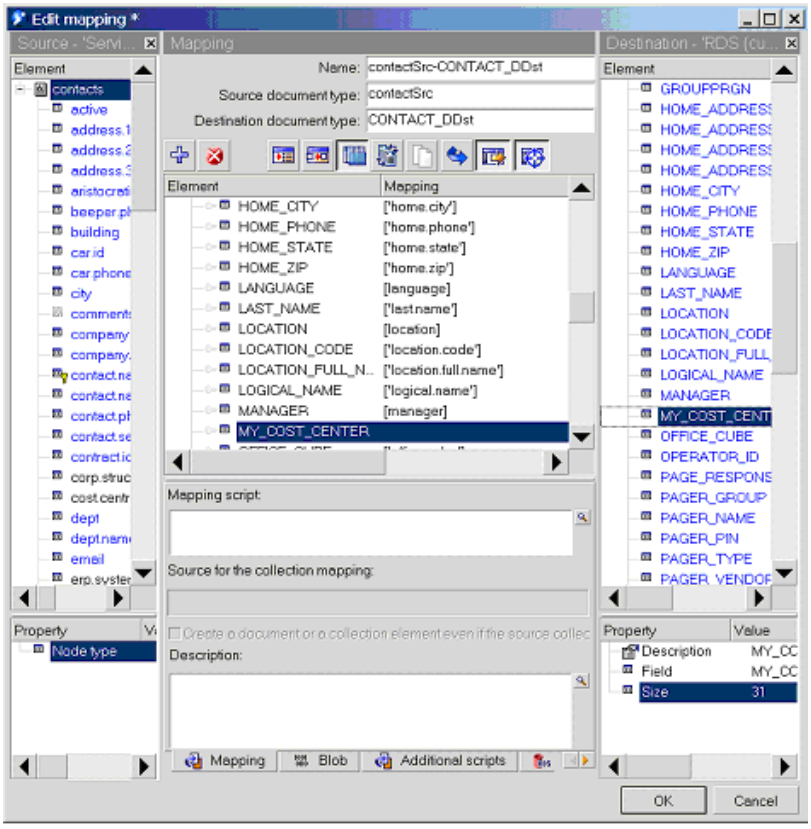
c Click the **Edit** button to open the detail edit mapping window.



9 Bind **MY_COST_CENTER** field in CONTACT_D in the RDS and **mycost.center** the *contacts* table in ServiceCenter.

a Select MY_COST_CENTER field from CONTACT_D element list box, on the right-hand side of the window.

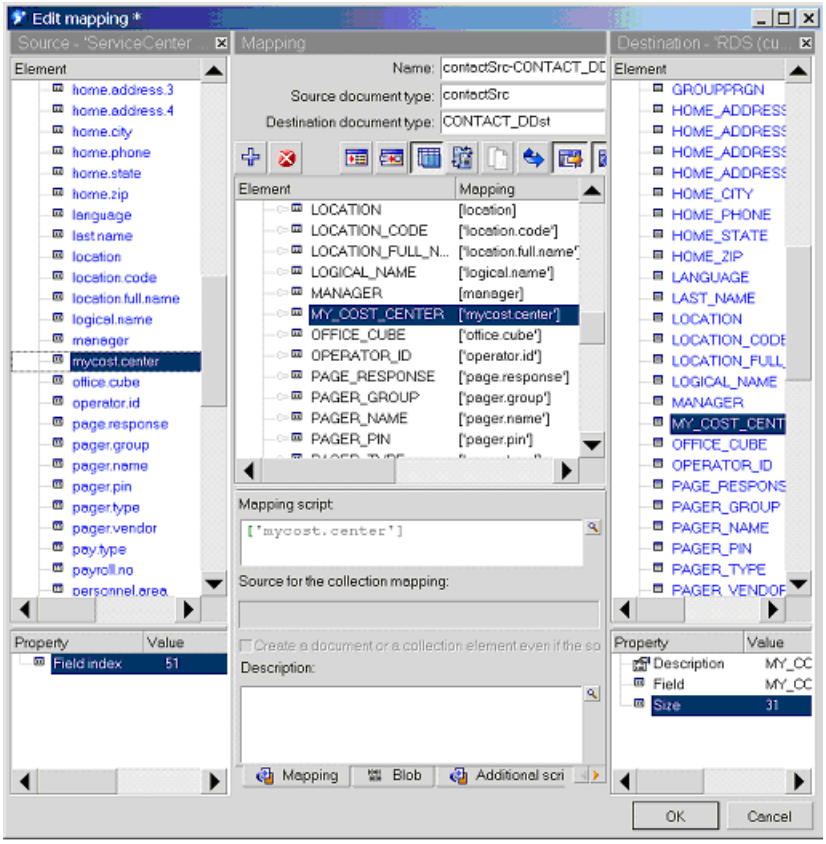
b Click the Add "+" button to add it to the Mapping element list box.



c Select the **mycost.center** field from contacts source element list.

d Drag-and-drop the **mycost.center** field on the destination element, MY_COST_CENTER.

e This completes the mapping of **mycost.center** of ServiceCenter *contacts* table to **my_cost_center** of the RDS CONTACT table. Click the **OK** button to save the new mapping information.



Scheduling automatic data synchronization

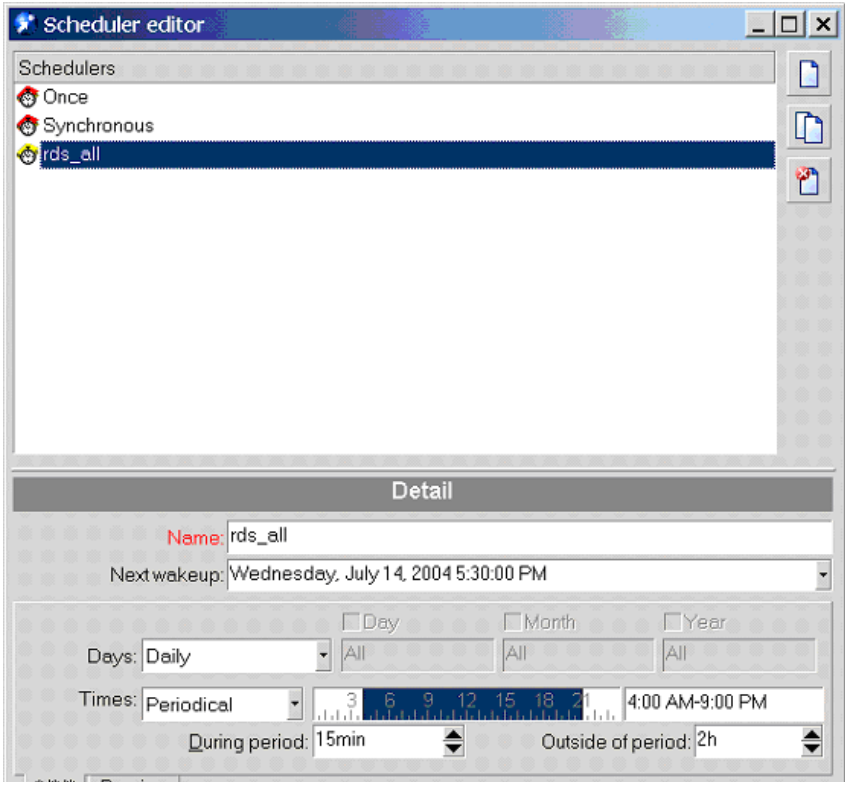
The RDS has set of pre-defined Connect-It scenario schedulers to run different synchronization tasks.

Because some of the synchronization tasks require more system resources than others, Peregrine recommends that they not be re-configured to occur more frequently than the default time intervals.

To schedule synchronization, you use the Connect-It Scheduler Editor.

To open the Connect-It Scheduler Editor

1. Click **Start** > **Programs** > **Peregrine** > **Connect-It** > **Service Console**.
2. Select the **rds_sc** scenario.
3. Click **Scheduling** to open the Connect-It Scheduling window.
4. Click **Edit Schedulers** button to open the Connect-It Scheduler Editor.



The following defines a synchronization schedule.

Scheduler	Description
rds_all	Synchronizes new and updated records at the default interval of once a day at midnight.

For more information about using the Connect-It Scheduler Editor, see the Connect-It documentation.

The rds_user interval can be changed using these same steps. However, if you change the rds_user interval in the Connect-It scenario, the BI Portal Administrative setting for "BIP user sync interval" must also be changed to match the rds_user interval. You can change the "BIP user sync interval" using BI Administration in the Administration module of the portal. Refer to *Using the BI Portal Administration page* (step 17) in the *BI Portal Administration Guide*, Chapter 4.

CHAPTER 6 Business Objects Universe Customization

This chapter explains how to use the Business Objects Designer tool to customize a universe file. The topics covered in this chapter are:

Editing the universe

Once you have customized the Reporting Data Store (RDS), you need to modify the metadata layer to expose this data to your end user. You do this by using the Business Objects Designer tool to customize the rds.unv file. The examples used in this section explain adding a new field, adding a new table, and modifying an existing field. You then need to check the integrity of the data and export the rds.unv file to the Business Objects server.

Adding a new table

This section provides an example of how to add a new table a universe file (rds.unv).

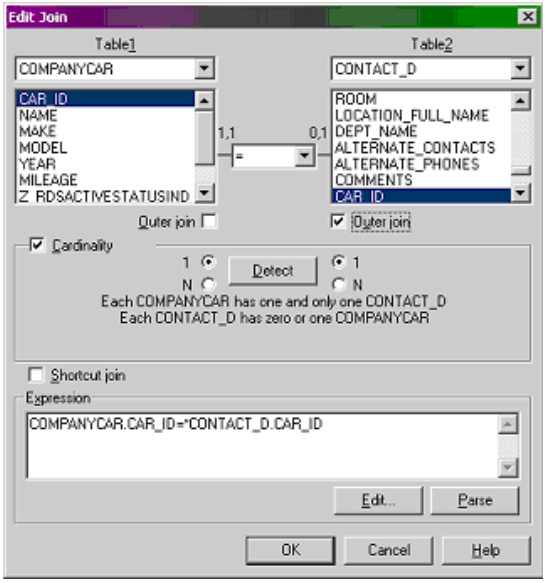
To add a new table

1. Click **Insert** > **Table**.
2. The Table Browser dialog box opens.
3. Select the **COMPANYCAR** table and then click **Insert**.
4. Click Close to close the window.

   The table now appears in the right pane of the main window as a table object.

   Next, you need to link this table to your existing tables. This will allow reports to use multiple tables successfully.
5. Click **Insert** > **Join**.

This opens the join dialog box.



6 In Table 1, select the first table of the join from the drop-down list and then click the field that used to link this table from the list. In our example, we will be linking from the **COMPANYCAR** table on the field **CAR_ID**.

7 In Table 2, select the RDS table and the field used to link to this table. In this example, it is **CONTACT_D**, using the field **CAR_ID**.

8 Between the two tables, there is a drop-down list box that specifies the qualifier for the join. If you are not sure which to use, default to equals (=).

9 Beneath each table, there is a check box for an Outer Join. Outer Joins are used when you want your query to return all record from the Outer table, regardless of whether or not there is a matching record in the other table. Inner joins only return records when there is a match in both tables. Since it is possible for a contact to exist without a company car, use an outer join on the contact side for this example. Click the appropriate check box.

   For more information on Inner vs. Outer Joins, consult your DBA or database documentation.

10 Click **Detect** to determine the Cardinality of your join then read the text that appears beneath the button that describes the join. If this does not sound correct, change the Cardinality as appropriate.

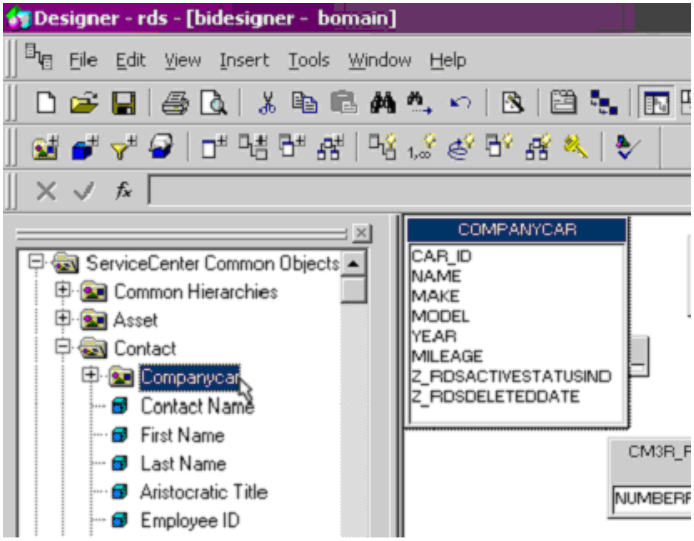
11 When you are satisfied with all the details of the link, click OK.

You now see a line drawn form COMPANYCAR to CONTACT_D. you may right-click on this line and select Join Properties at any time to edit the settings again.

You now need to add this table to the object browser (right pane). This makes the fields available to be selected for a report. This can be done individually, or all at once. For this example, create a folder under the Contacts folder.

To add this table to the object browser

1 Expand the folders until you have opened the **ServiceCenter Common Objects** > **Contact** folder.



2 Click and drag the **COMPANYCAR** table from the left pane to the Contact folder.

This automatically creates a folder containing all the fields in that table.

3 To rename any object, right-click it and select Class Properties (for folders) or Object Properties (for fields). Edit the name or description and click **OK**.

Note: Peregrine recommends as a best practice that you include a common string of text in the descriptions of all universe objects you add or edit, so that a quick search can be used to identify them during future upgrades. For example, the description for Car ID above could be "Unique identifier that links a contact to a company car (CUSTOM OBJECT)". In the future, a quick search on "(CUSTOM OBJECT)" in the description would reveal all personalized items.

4 To hide any field from end-users, right-click the object and select Hide Item(s).

5 To move any object, click and drag it to the desired folder.

Adding a new field

This section provides an example of how to add a new field a universe file (rds.unv).

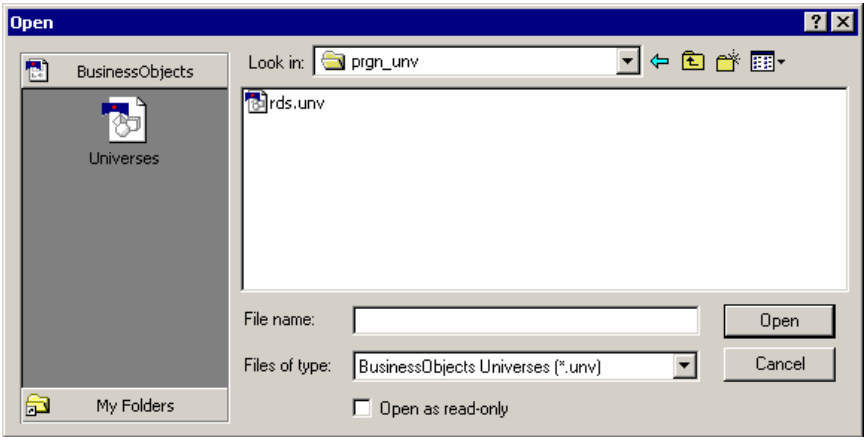
To edit the rds.unv file

1 Log on to your Business Objects Designer Tool as a user with Designer rights.

2 Make a copy of your universe file (rds.unv). This can be found in the \nodes\<server>\<cluster>\universes\<domain>\ directory.

Note: This copy is in addition to the backup copy you made earlier in the customization workflow. This is the copy you will edit. Do not edit the live copy.

3 In the Designer window, click **File** > **Open** and navigate to the universe file (rds.unv).



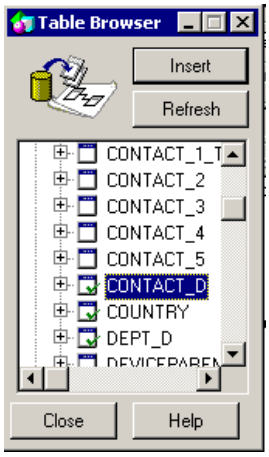
4 In the Open dialog box, select the universe file (**rds.unv**) and click **Open**.

In this example, you will be adding one new field to the CONTACT_D table.

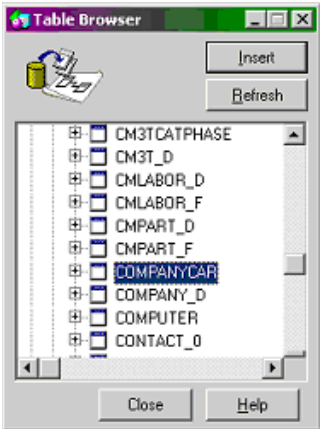
To add a new field

1 In the Designer window, click **Insert** > **Table**.

The Table Browser dialog box opens and displays a list of all tables in the RDS.



2 Click the Plus sign (+) next to CONTACT_D to expand the table and see individual fields.



3 Click the new field (in this case: CAR_ID) and then click **Insert** to add the field to the universe.

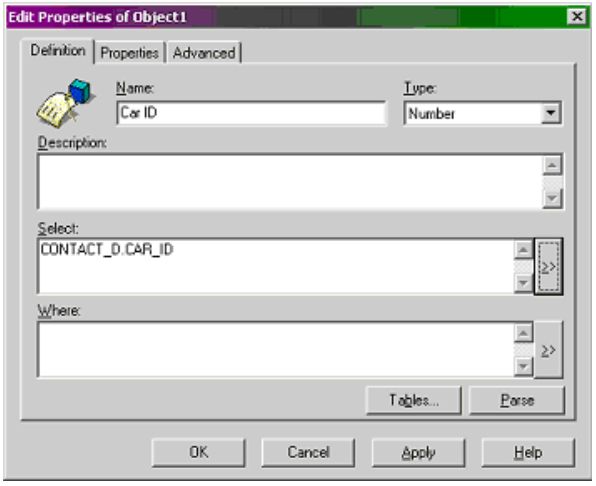
4 Click **Close** to close the window.

Next, you need to add this field to the object browser (right pane). This makes the field available to be selected for a report.

To add this field to the object browser

1 In the Designer window, expand the folders until you have opened the **ServiceCenter Common Objects** > **Contact** folder.

2 Right-click the **Contact** folder and select **Object**.



3 In the Edit Properties of Object window, type a name in the Name text box, for example Car ID.

4 In the Type drop-down list box, select a type, for example, Number.

5 Type a description for the field in the Description text box.

Note: Peregrine recommend that as a best practice you include a common string of text in the descriptions of all universe objects you add or edit, so that a quick search can be used to identify them during future upgrades. For example, the description for Car ID above could be "Unique identifier that links a contact to a company car (CUSTOM OBJECT)". In the future, a quick search on "(CUSTOM OBJECT)" in the description would reveal all personalized items.

6 Select the appropriate data type in the Type drop-down list box.

7 In the Select text box, either browse for the field by clicking the >> button, or type in the field in the form <**tablename**>.<**fieldname**>.

8 If necessary, you can also add filters in the Where text box.

9 Test your object by clicking **Parse**.

10 If the test is successful, click **OK** to apply changes and close this dialog. Otherwise, re-enter any erroneous entries.

11 If you want, you can now click and drag the new object to the appropriate folder / location.

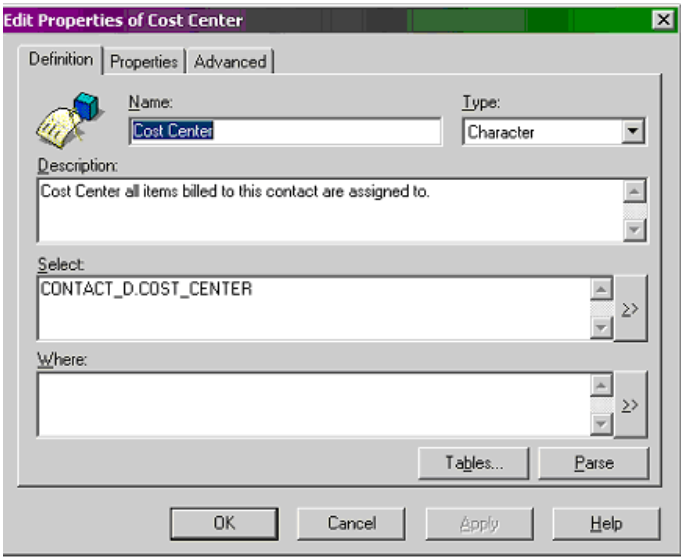
Modifying an existing field

This section provides an example of how to modify an existing field a universe file (rds.unv).

To modify an existing field

1 Browse to the field in the left pane.

2 Right-click and select Object Properties.



3 In the Edit Properties dialog box, edit the name, data type, description, and database field as needed.

4 In our example, only the name needs to be altered. Change Cost Center to My Cost Center.

5 Click **Parse** to test the changes.

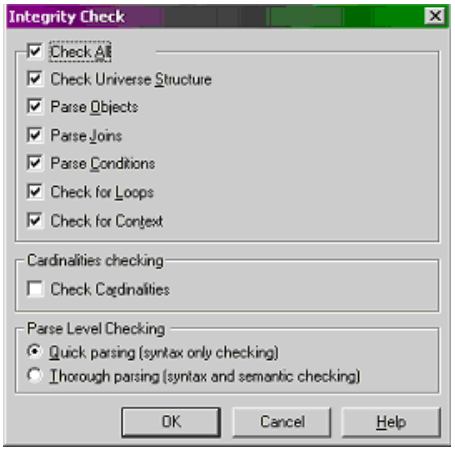
6 If successful, click **OK** to save the changes; otherwise, make the necessary changes in the dialog box and repeat the test.

Checking database integrity

Once you've made all your changes, you need to do an integrity check to make sure all database links are valid.

To do an integrity check

1 Click **Tools** > **Check Integrity**.



2 Click **Check All**.

3 Click **OK**.

The utility runs though the Universe and suggest possible problem areas.

4 Correct any problems as needed.

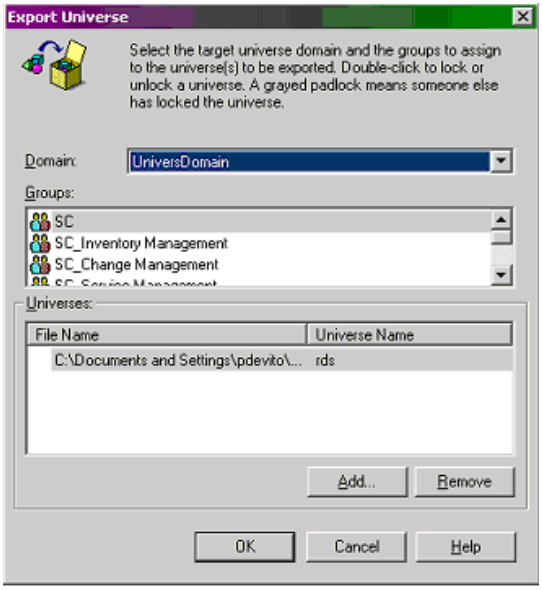
5 Save your changes.

Exporting the universe

Once you have completed your changes and verified the integrity of the database you are ready to export the changes to the Business Objects server.

To export the universe to the Business Objects server

1 Click **File** > **Export**.



2 In the Export Universe dialog box, click the appropriate Domain from the Domain drop-down list.

3 In the Groups list, select a Group.

Note: The universe should be exported to the SC application group.

4 Click OK to export the universe.

After a short period, a message indicating successful export appears.

Note: If you have altered any existing objects, Peregrine recommends that you check any existing reports on your server that may reference those objects. You may need to edit the query to reflect your new changes.

Editing security profiles

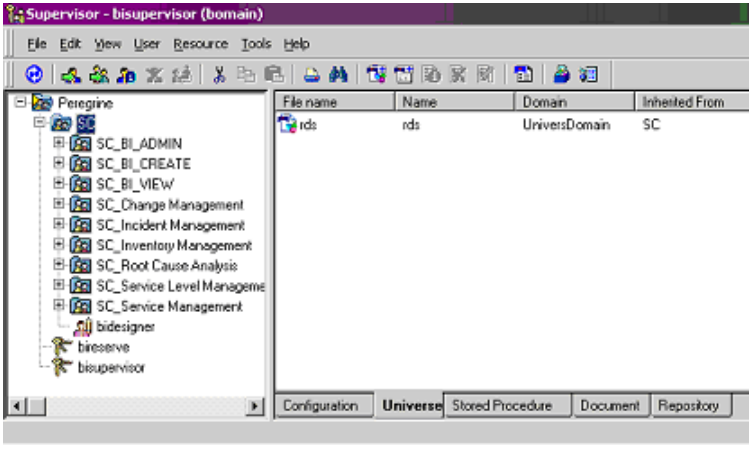
This is an optional procedure and only required if you have set up row level security in Business Objects. For additional information, see the *BI Portal Administration Guide*.

To edit a security profile

1. Log on to the Business Objects Supervisor as a user with Supervisor rights.
2. Select the group at the level you want to set your restrictions. If you want to restrict all users, this would be your main application group. If you want to impose different restrictions for different sets of users,
   a. Create groups for specific restrictions under the application group.
   b. Add users to groups you created.

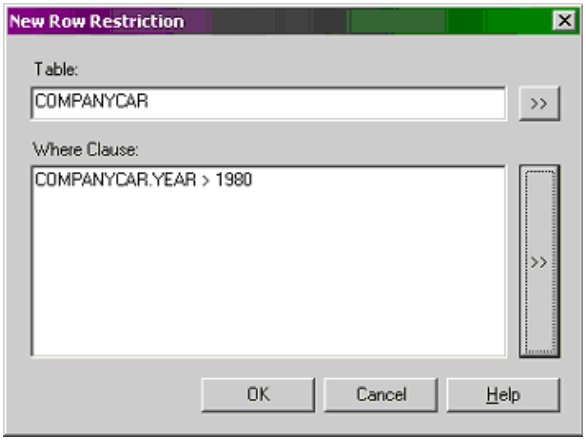
To set security restrictions on the sub-group

1. Highlight the sub-group.
2. From the Supervisor window, click the **Universe** tab at the bottom of the window.



3. On the Universe tab, right-click the **rds** universe file and select **Properties**.
4. Click the **Rows** tab.

5 Click **Add** to create a new restriction.



6 Either type in a table name or click the >> key next to the Table field to browse for the table you want to add restrictions to. This example uses COMPANYCAR.

7 In the Where Clause field, either type in a SQL clause or click the >> key field to build one with the formula editor. For help with SQL syntax, contact your DBA or database documentation.

8 This example, restricts access to all records in the COMPANYCAR table that refer to cars built before 1980 (COMPANYCAR.YEAR > 1980).

9 Click **OK** when done.

10 Click **Check All** to verify the validity of the SQL you entered.

The Status field should change to read OK.

11 Click **OK** to close this dialog box.

Note: Changes are instantaneous. You do not need to save or reset the server for alterations to take effect.

To test the restriction

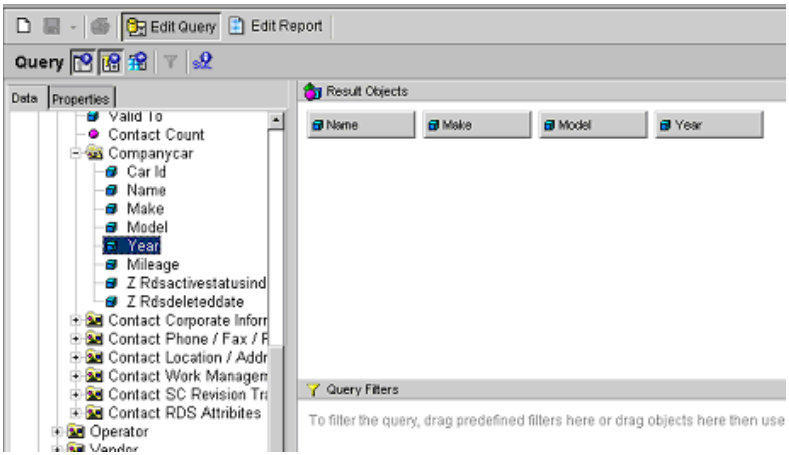
1 Log on to BI Portal as a user with report creation rights.

Note: Make sure this user belongs to the group that you placed the restriction on.

2 Click the **Reporting** icon.

3 Click **Create**.

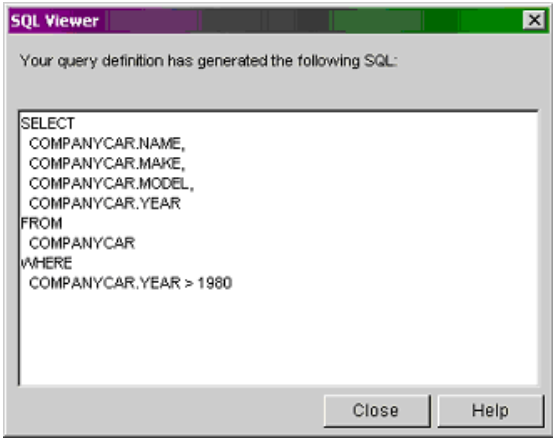
4 Click the rds universe to launch the Report Designer tool.



5 Navigate to the appropriate object in the universe and drag fields into the Result Objects pane.

6 Click the **View SQL** button to see the SQL statement behind the report.

You should see that the restrictive clause you added has been tacked on the the report.



To add additional access restrictions, repeat the steps in this section, as required.

APPENDIX A RDS System Tables

This appendix provides descriptions and samples of the system tables found in the RDS. The topics covered are:

Tracking RDS schema metadata

The RDS tables used to track RDS schema metadata include:

- RDS_UNIQUECOLUMN table
- RDS_SCDCOLUMN table

RDS_UNIQUECOLUMN table

The RDS_UNIQUECOLUMN table tracks all the unique columns for the reporting tables. It also includes unique fields for the ServiceCenter source files. The RDS schema creation program uses this information to create unique indexes for RDS reporting tables. The RDS Connect-It scenarios use the unique field values to decide about adding new records or updating the existing records in the ETL process.

For an Oracle database, the RDS_UNIQUECOLUMN table columns are defined as follows.

Name	Null?	Type	Description
LRDS_UNIQUECOLUMN_ID	Not NULL	NUMBER(10)	Unique system key.
TABLE_NAME	Not NULL	VARCHAR2(80)	RDS Report table name. RDS table type as one of following values: ASSOCIATE DIMENSION
TABLE_TYPE	Not NULL	VARCHAR2(80)	DIRECT_MAPPING
DS_TABLE_NAME	Not NULL	VARCHAR2(30)	Source file name. File type as on the following values: DATA_SOURCE (as SC source file) PARENT_DIM Note: PARENT_DIM information is used for cascade delete. When the record in the table, DS_TABLE_NAME is deleted, the record of the table, TABLE_NAME if
SRC_UNIQUECOLUMN	Not NULL	VARCHAR2(80)	
SRC_UNIQUECOLUMN	Not NULL	VARCHAR2(80)	
RDS_UNIQUECOLUMN	Not NULL	VARCHAR2(30)	
TABLE_UNIQUECOLUMN	Not NULL	VARCHAR2(10)	

RDS_SCDCOLUMN table

The **RDS_SCDCOLUMN** table tracks all the SCD columns for the dimension tables. The RDS Connect-It scenario uses this information to decide which updating to use; the updating as simple overwrite updating or SCD Type Two updating in the ETL process. The SCD Type Two updating creates a new record and marks the existing record as old in dimension tables. More information about SCD Type Two can be found in different data warehouse technical sources.

For an Oracle database, the RDS_SCDCOLUMN table columns are defined as follows:

Name	Null?	Type	Description
LRDS_SCDCOLUMN_ID	Not NULL	NUMBER(10)	Unique system key.
TABLE_NAME	Not NULL	VARCHAR2(80)	RDS dimension table name.
SRC_SCDCOLUMN	Not NULL	VARCHAR2(80)	ServiceCenter source field name.
RDS_SCDCOLUMN	Not NULL	VARCHAR2(80)	RDS column name.

Name	Null?	Type	Description
TYPE_SCDCOLUMN	Not NULL	VARCHAR2(30)	Column type.
TABLE_SCDNUMBER	Not NULL	NUMBER(10)	The sequence number of the SCD column if there are more than one SCD columns.

RDS environment tables

This section describes the tables that store data about the RDS environment.

RDS_DBINFO table

The **RDS_DBINFO** table stores the RDS deployment environment. The following table shows sample data.

Z_RDS_DBINFO_ID	OSTYPE	DATABASETYPE	DATABASENAME	DATABASEURL	USERID	USERPASSWORD
1	windows	Oracle		jdbc:oracle:oci8:@rds	rds_dba	passw0rd

RDS Security System Table - RDS_USER

The RDS_USER table stores all the user information for BI Portal. The data source for RDS_USER can be the ServiceCenter *operator* table and *contacts* table if contact based authentication is turned on.

The RDS_User table is defined as follows.

Name	Null?	Type	Description
RDS_USER_ID		FLOAT	Unique system key.
DS_UNIKEY_NAME		VARCHAR2(120)	The unique key value. For the operator file, the value is the operator name field value.
CAPABILITIES		VARCHAR2(500)	
APP_NAME		VARCHAR2(10)	
DS_TYPE		VARCHAR2(30)	
EMPLID		VARCHAR2(140)	The join field value for contact with operator.
ROLEGROUP_ID		FLOAT	RDS_USER_ID from the operator record, which is linked with the contact.
Z_RDSLASTMODDATE		DATE	Time the record was last modified.

Name	Null?	Type	Description
Z_RDSACTIVESTATUSIND		VARCHAR2(1)	The user record active status; 'Y' or 'N'.
Z_RDSDELETEDATE		DATE	Record deleted date time.

RDS System tables for tracking ETL processing

The tables used to track RDS ETL processing status are:

- RDS_CIT
- RDS_CIT_LOG
- RDS_ETL_WORK
- RDS_ETLSYNC_LOG
- RDS_SEC_SYNC
- RDS_SEQUENCE

The RDS_CIT and RDS_ETL_WORK tables are used for RDS ETL (Extracting, Transforming and loading) process as RDS internal tracking system tables.

RDS_CIT_LOG

RDS_CIT_LOG has the columns of:

- ID
- STATE_FLAG
- CIT_TIME

RDS_CIT_LOG STATE_FLAG

The **RDS_CIT_LOG STATE_FLAG** shows the ETL process status as one of the following values:

- CIT_START - RDS Connect-It scenario initial sync or periodic sync starts.
- RDS_SYNC_START - RDS fact tables population starts.
- RDS_AGG_START - RDS aggregation tables population starts.
- RDS_HIER_START - RDS hierarchy tables population starts.
- COMPLETED - ETL process is completed.

The STATE_FLAG shows COMPLETED after ETL process is done.

RDS_SEC_SYNC table

The **RDS_SEC_SYNC** table records the last time a different BI Portal user performed a synchronization.

RDS_SEQUENCE table

The **RDS_SEQUENCE** table stores the next available sequence value for the unique system keys for all the RDS tables.

RDS_ETLSYNC_LOG table

The **RDS_ETLSYNC_LOG** provides very useful information for the ETL sync volume. This table is defined as:

Name	Null?	Type	Description
RDS_ETLSYNC_LOG_ID	NOT NULL	NUMBER	Unique system key.
TABLE_NAME	NOT NULL	VARCHAR2(80)	Populated table name.
TABLE_ACTION	NOT NULL	VARCHAR2(80)	ETL process updating action codes: Add Update Update with SCD Delete Log
SYNC_RECORD_COUNT	NOT NULL	NUMBER	Total record number of the updating described by the action code.
SYNC_DATETIME		DATE	The sync date time
Z_RDSCREATEDDATE		DATE	The track record creation date time

Other useful tables

This section describes other system tables that provide useful information.

RDS time dimension table

The R**DS time dimension** table is a unique and powerful dimension in the RDS. The RDS time dimension table provides the mechanism for BI Portal reporting based on days, months, quarters and years. All RDS aggregation tables ended with '_AGG'. Table name convention are based on weeks, quarters and months, and years. RDS_TIMEDIM_D stores all the calendar day from 1995 to 2010. The start year and end year values are set by RDS installer. The values are saved in RDS \conf directory, rds.properties file as:

```
rds.startDate=1995
rds.endDate=2010
```

The RDS time dimension table columns are defined as follows:

Name	Null?	Type
RDS_TIMEDIM_DID	Not NULL	NUMBER
FULLDATE		VARCHAR2(20)
WEEK		VARCHAR2(10)
MONTH		VARCHAR2(20)
QUARTER		VARCHAR2(4)
YEAR		VARCHAR2(10)
FISCALPERIOD		VARCHAR2(4)
LASTDAYINMONTHFLAG		CHAR(1)
FULLDATE_D		DATE
MONTH_NUMERIC		NUMBER

Index


Peregrine
SYSTEMS