Configuration Deployment Tool Guide

Sterling Commerce

An IBM Company

Contents

1

Deployment Tools Overview

During incremental configurations, changes are typically developed in a test environment and then they are rolled out into a production environment. Migrating configuration data can be fairly cumbersome and time consuming. The Configuration Deployment Tool enables you to migrate configuration data. This tool ensures data integrity while reducing the system downtime to transfer data and minimizing the effort needed to ensure accuracy.

This tool is designed to migrate data that is modified as part of a normal day-to-day operation. Note that the Configuration Deployment Tool can be used to deploy configuration data that is the result of an upgrade, but it should not be used to perform the data upgrade itself.

Configuration Deployment Tool Features

The Configuration Deployment Tool (CDT) can be accessed from the Sterling Development and Deployment WorkBench (also known as the "WorkBench").

CDT provides the following capabilities:

- Transfer complete and partial sets of configuration data or discrete logical portions.
- Transfer data to and from XML files or databases.
- Transform certain data-like IP addresses and port numbers that are different in two environments, depending upon network configuration.
- Generate a report of configuration differences by comparing the two systems.

2

Concepts

This chapter presents and defines terms relevant to the Configuration Deployment Tool, and explains how they apply.

2.1 Source and Target Environments

The Configuration Deployment Tool deploys data from one environment to another. The deployment could occur from development to test environments, from staging to production environments, and so forth. The environment that serves as the point of origin for the data is known as the "source" environment. The destination environment into which data is deployed is defined as the "target" environment. This deployment can be in the form of importing and exporting data to and from databases or XML files.

Note: If you make any changes to the configuration data in the source database, the existing transactions in the CDT may get affected.

2.2 Configuration Groups and Driver Entities

The entire set of configuration data is broken down into logical subsets called "configuration groups" and "driver entities". Configuration groups and driver entities are predefined and cannot be changed.

During the deployment process, if you need to perform more granular inserts, updates, and deletes so that your target database matches your source, you choose these configuration groups or driver entities.

Driver Entities

Most configuration data can be deployed starting with a logical entity, for example, an organization or a pipeline. These logical entities are called "driver entities". Driver entities represent the most granular level of information that can be deployed from the source to the target without loss of data integrity.

Only driver entities allow deployment at a record level. For other tables either of the following conditions apply:

- The table is completely deployed if it is not dependent on any driver entity.
- Only records corresponding to the driver entity are deployed.

Information about driver entities can be stored in multiple tables and when deploying an entity, data in all related tables is deployed together in one transaction boundary to preserve data consistency.

Configuration Groups

Logically related tables or driver entities are also grouped together into "configuration groups" that typically represent larger, significant logical data models. Examples include the Business Process Model or the Participant Model. These groups are provided for convenience and for ease of navigation on the user interface.

2.3 Externally Maintained Configuration Data

In your implementation, you may be required to import certain data into your target that is not part of your source database. For these tables, you should not use CDT to deploy data as it does not have access to the correct data.

Best Practices

If you must use the Configuration Deployment Tool to deploy externally maintained data, the recommended way to handle this is to import this data into the source and then use CDT to deploy it into the target. This guarantees data integrity.

If you cannot import this data into your source database, you can work with external data by ensuring that the target database either ignores these tables or appends them. Use the Ignore and Append-only features only if you have tried all other available options and only after subjecting your environment to rigorous testing.

Caution: When using the Ignore or Append-only features, CDT cannot guarantee the integrity of any external data. In order to ensure data integrity, CDT must have complete access to the configuration data.

Ignore

In cases where data in tables is maintained externally, you can omit these tables from the deployment operation by specifying a preference for them to be ignored.

Ignoring a table or a driver entity also automatically ignores all its dependent tables. However, there are some tables that store data for multiple driver entities and are present in multiple groups.An example of this is the YFS_GRAPH_UI table that contains data for pipelines, services and statuses. Ignoring one of these tables causes CDT to incorrectly mark the corresponding records for deletion.

Append-only

In cases where some tables are partially maintained externally, you can specify preferences to ensure that these tables are deployed in an "append-only" mode.

For append-only tables, the dependent tables are not ignored. Marking a table as append-only implies that only a few rows in the target database are maintained on the source system—other rows are externally imported. In such cases, it is extremely important that there is no overlap between the data present in the source and the external system.For example, if you maintain your shipping nodes in the source database and import store information directly into the target, you must not have any stores in the source database. This leads to unpredictable results.

2.4 Deploying Custom Tables

CDT automatically deploys configuration tables and extensions defined within the database framework. If you have custom configuration tables defined in your installation, CDT needs to be specially configured to deploy these tables. To enable CDT to deploy these tables, the tables need to be registered with CDT by creating a special custom deployment XML file, called `cdt_custom.xml`.

A sample of this file can be found in your `<INSTALL_DIR>/resources/ydkresources<INSTALL_DIR>/resources/ydkresources` directory. This file defines a group named "Custom Tables" and should include a list of your custom tables. CDT automatically compares, displays and deploys changes to custom records for all tables that have one or more primary key columns.

To deploy custom tables:

1. Place your custom xml file (e.g., fsa_additions.xml) that contains the custom entity definitions in the `<INSTALL_DIR>`/repository/entity/pcaxmls/ directory.
2. Change directory to `<INSTALL_DIR>`/bin
3. Run, ./ant.sh -Dtable=<*TABLE_NAME*> -f templateXmlGen.xml
4. You can now see an `extn` folder created under `<INSTALL_DIR>/` directory that contains custom api template xmls.
5. From `<INSTALL_DIR>`/bin run, ./dbverify.sh
6. From `<INSTALL_DIR>`/bin run, ./deployer.sh -t entitydeployer -l info

Update the Runtime for these modifications.

This tool does not support custom tables as drivers or the representation of custom tables in a dependency tree structure. As a result, all custom tables can only be deployed together as part of the "Custom Tables" group. It also does not support custom tables without a primary key.

The `cdt_custom.xml` file contains the following:

```
<Group Name="Custom Tables">
    <Table Name="CUSTOM_CONFIG_TABLE_1"/>
    <Table Name="CUSTOM_CONFIG_TABLE_2"/>
</Group>
```

2.5 Users/Permissions/Roles Tables

Users/permissions/roles tables can be exported via the CDT UI. However, these types of tables need to be done in a separate export as they cannot be exported with other resources.

2.6 Foreign Key Checks

The CDT enforces data consistency by deploying all related tables that define an entity together in one operation. In addition, to ensure data integrity, the CDT also checks the required foreign key constraints for each table - which could potentially be defined for a table in a completely different group. Therefore, when deploying a small subset of data, it is possible that you may see error messages indicating foreign key constraint violations if the corresponding data in the independent table is not being deployed in the same operation. In this case, you should try deploying a bigger set of data. Note that foreign key constraints are not defined or checked for custom tables.

To provide the best performance, foreign key constraints are not checked when deploying the complete configuration.

2.7 Data Transformations

Frequently, the development and production environments have different values for network settings such as server names and IP addresses. Some configuration data tables store host names, IP addresses, and URLs. While these are valid for your source environment, when deploying this data into the target environment, the configuration must be updated with the corresponding values applicable to the target environment. The CDT enables you to automatically transform these data elements into target-appropriate values by letting you specify transformations to be carried out on the source data *before* it is deployed into the target.

2.8 Business Processes

Business Processes re-imported into another system always need to be enabled manually to ensure they function correctly in the new environment. Additionally, any adapters or schedules related to the business processes must also be enabled manually.

3

Understanding the CDT Interface

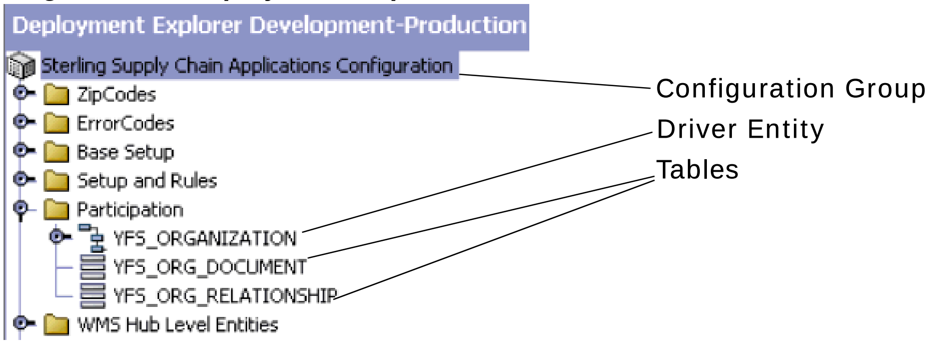
When the Configuration Deployment Tool starts, it prompts you to specify the details about your source and target databases to use during the session. After you have successfully connected to your source and target databases, the Deployment Explorer window appears.

3.1 The Deployment Explorer

The Deployment Explorer window displays the list of configuration groups, driver entities, and tables that can be deployed. The names you define for the source and target databases are displayed in the heading panel.

Each time you log into the Configuration Deployment Tool there is one instance of this window.

Figure 3–1 Deployment Explorer Window



Deployment Explorer Development-Production
- Sterling Supply Chain Applications Configuration
  - ZipCodes
  - ErrorCodes
  - Base Setup
  - Setup and Rules
  - Participation
    - YFS_ORGANIZATION
    - YFS_ORG_DOCUMENT
    - YFS_ORG_RELATIONSHIP
  - WMS Hub Level Entities

You can choose the configuration group or the driver entity that you want to compare between the source and target databases.

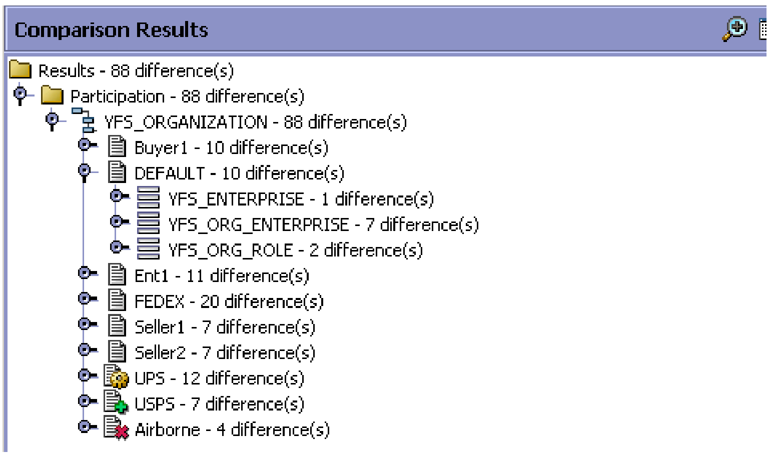
During the compare operation, the progress and the results of the comparison operation are displayed in the Comparison Results window and in the Status panel.

3.2 Comparison Results Window

The Comparison Results window displays the outcome of the comparison between the source and target databases.

The Comparison Results window displays information pertaining to the current session. Only one Comparison Results window can be displayed during each session. After viewing the results of one comparison, you must close the window before you can compare a different set of tables.

Figure 3–2 Comparison Results Window



Comparison Results

Results - 88 difference(s)
Participation - 88 difference(s)
YFS_ORGANIZATION - 88 difference(s)
Buyer1 - 10 difference(s)
DEFAULT - 10 difference(s)
YFS_ENTERPRISE - 1 difference(s)
YFS_ORG_ENTERPRISE - 7 difference(s)
YFS_ORG_ROLE - 2 difference(s)
Ent1 - 11 difference(s)
FEDEX - 20 difference(s)
Seller1 - 7 difference(s)
Seller2 - 7 difference(s)
UPS - 12 difference(s)
USPS - 7 difference(s)
Airborne - 4 difference(s)

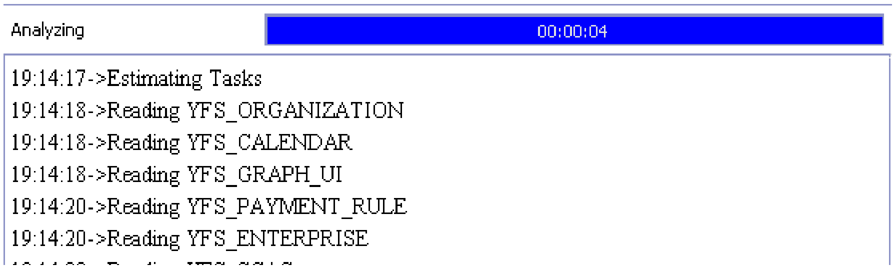
After generating comparison results, you can carry out any one of the following tasks:

- Generate a report of the differences
- View the details of each difference
- Deploy configuration data from the source database into the target database

3.3 The Deployment Tool Status Panel

The Status panel displays information about operations while they are carried out.

Figure 3–3 Configuration Deployment Tool Status Panel



```
19:14:17->Estimating Tasks
19:14:18->Reading YFS_ORGANIZATION
19:14:18->Reading YFS_CALENDAR
19:14:18->Reading YFS_GRAPH_UI
19:14:20->Reading YFS_PAYMENT_RULE
19:14:20->Reading YFS_ENTERPRISE
```

4

Before Using the Configuration Deployment Tool

Before you begin using the Configuration Deployment Tool, ensure that you have addressed the policies and system requirements described in this section.

Also, to make sure that all tables appear in the Configuration Deployment Tool, copy the file specified in your application `<application>-config-db.xml` file in the `<INSTALL_DIR>/database/cdt` directory to `config-db.xml` in the same directory.

Config-db.xml File

The config-db.xml file is the configuration file used by the CDT to display the logical or functional grouping of tables into which data is to be deployed. This file organizes the tables in a hierarchical manner based upon the functional dependencies among them. The actual data deployment is carried out in accordance with this configuration or hierarchy. The config-db.xml file is located in the `<INSTALL_DIR>/database/cdt/` directory.

4.1 System Requirements

The RAM requirements of the Configuration Deployment Tool depend on the size of your database and the distribution of your configuration data.

4.1.1 Time Estimates

The time required for the Configuration Deployment Tool to perform comparison and deployment tasks varies according to your system resources and the size and distribution of your configuration data. For example, processing time may increase when there are many records in a table that are referenced by foreign key constraints from other tables, or when there are many records in a table that serves as a driver entity.

During tests of the Configuration Deployment Tool, the amount of time it took to perform tasks on a Pentium class machine with 512 MB of RAM and running at 550 MHz was measured. The Configuration Deployment Tool performed as described in Table 4–1, "Time Estimates using the Configuration Deployment Tool".

Table 4–1 Time Estimates using the Configuration Deployment Tool

Task	Description of Databases	Time
Comparison	Source database - 110,000 records Target database - 110,000 records	7 minutes
Comparison	Source database - 110,000 records Target database - empty	4 minutes
Deployment	110,000 differences	11 minutes

4.1.2 Installation

The Configuration Deployment Tool is installed automatically during the installation.

4.1.3 Environment State

The Configuration Deployment Tool assumes that your source and target environments match exactly in the following respects:

- Release of the application being deployed (including hot fixes)
- Release of JDBC drivers
- Release of database software
- Database structure (schema objects such as tables, indexes, and sequences)

As this Configuration Deployment Tool is used by technical professionals for tasks they perform on an occasional basis, it is **not** localizable or customizable. However, you can specify configuration preferences as described in this chapter.

4.2 Security Strategy

The Configuration Deployment Tool makes use of the user authentication and authorization supplied by your database provider. Access control and authorization are not specified through the Configuration Deployment Tool.

Ensure that the person using the Configuration Deployment Tool has sufficient authentication privileges (select, insert, update, and delete) for both databases; full DBA privileges are not required.

4.3 CDT Change Management Strategy

The Configuration Deployment Tool does **not** enforce checks to restrict configuration data modification on the source or the target schemas using other means. You must develop and enforce your own methodology.

For example, if you use the Configuration Deployment Tool to migrate data from staging to production, it is **not** expected that the configuration in production is modified by means other than this tool. In such a case those changes are overwritten the next time CDT is run. Also, this could potentially lead to data integrity issues if the changes are performed in either the source or target while CDT is being run.

The Configuration Deployment Tool is **not** supported for implementations where configuration data is directly modified in production using the Configurator or any other means. For exceptional cases like urgent or critical fixes to configuration data in production, you must update the staging database with the same changes.

4.4 Rollback Strategy

To prevent application failure and downtime, implement a rigorous rollback methodology that involves creating a backup snapshot of your configuration data in production before you use the Configuration Deployment Tool to deploy changes. This backup is accomplished by using the database-specific export and import utilities. The CDT provides some samples for Oracle, DB2, and SQL Server databases that you can customize for your own use.

4.5 Upgrades and Maintenance

Using the Configuration Deployment Tool should not impact the methodology for applying upgrades or hot fixes in a multi-step staging environment.

The upgrade methodology being followed should not change for environments already set up for staging before production. However, the Configuration Deployment Tool by itself does **not** provide support for all of the processes and methodologies required for supporting a multi-step application staging and deployment environment because it is only capable of deploying configuration data.

The process of applying product upgrades and patches is especially complex in an environment where the staging area must be kept synchronized with production. One way to keep these environments harmonious is to apply software patches to both systems simultaneously and reverse deploy the data upgrades. This is because application data upgrades may behave differently and produce different results based on the transactional data they encounter. If this application data upgrade is run independently on production and staging, the results may be significantly different as a result of the differences in transactional data that the upgrade program encountered. In such a case, the production snapshot should be treated as the baseline and reverse deployed into staging. This can be accomplished by configuring your production database as the source and your staging database as the target.

5 Setting Up and Running the CDT

Before running the Configuration Deployment Tool, you can set preferences for its use.

5.1 Setting Preference Settings

You can configure preferences (such as a reports directory) and parameters that determine the behavior of the comparison operation. When you modify these properties, the changes persist, so you do not need to reset them each time you use CDT. These changes are saved in the `<INSTALL_DIR>/resources/ydkresources/ydkprefs.xml` file.

Note: The ydkprefs.xml file is created the first time you start the CDT.

To specify Configuration Deployment Tool settings:

1. From the Deployment Explorer action bar, choose the *Preferences* icon.
2. In the Preferences window, fill in values using the descriptions provided in Table 5–1.

Table 5–1 Configuration Deployment Tool Preference Settings

Control	Description
Settings Tab	
Reports Directory	Specify the absolute path where you want reports to be generated.

Table 5–1 Configuration Deployment Tool Preference Settings

Control	**Description**
Custom Deployment Class	Specify the name of the class that should be invoked for deploying custom tables not handled by CDT.
Max Changes to Display	Specify the maximum number of differences to be displayed. The default display number is 100.
Transformations Tab	
Table Element	Tables that can be added or deleted.
Table Name Attribute	Specify the name of the table on which you want to carry out the transformation. The syntax and case must match the name of the table used in the ERDs. Custom tables cannot be transformed. Choose the Details icon to specify a value.
Column Element	Columns that can be added or deleted.
Column Name Attribute	Specify the name of the column containing the data to be transformed. The syntax and case must match the name of the column used in the ERDs. Extended columns can be transformed. Choose the Details icon to specify a value.
Transform Element	Define the transformation for this column. For each column, you can define one or more transformations. These transformations are applied to data in this column in sequential order. You can specify multiple transformations for each column, using the delete action to remove the parent element.
Match Attribute	Specify the pattern to search for in the source data. All matching occurrences of this pattern are replaced with the value specified in the Replace attribute. Choose the Details icon to specify a value.
Replace Attribute	Specify the value to replace the pattern with. Choose the Details icon to specify a value.

Table 5–1 Configuration Deployment Tool Preference Settings

Control	Description
XPath Attribute	Conditional. If the column to be transformed contains non-XML data, you do not need to specify this XPath attribute. However, some configuration information is stored as XML in the database. If the column to be transformed contains XML data, use this attribute to specify the location of the exact attribute to be transformed. Use the syntax: `xml:/Configuration/Connection/Host/@IPAddress`. Choose the Details icon to specify a value.
Append-only Tables Tab	
Append-only Tables	Specify any configuration tables in which *some* rows maintain external data. This prevents the data from being deleted during deployment. Specify that table and all of its dependent tables. **Note**: Rows that are maintained externally should never be present in your source database, since this can lead to unpredictable results.
Ignore Tables Tab	
Ignore Tables	Specify any external configuration tables that you do not want the tool to deploy from the source to the target. Ignoring a table automatically ignores all dependent tables as well.

5.2 Running the Configuration Deployment Tool

The Configuration Deployment Tool has an interactive graphical user interface and noninteractive command-line mode. The interactive interface is used to perform data comparison tasks, deploy data, and promote services. The command-line mode is used to schedule or run a data deployment. For more information on using the command-line mode, refer to Section 7.1.1, "Deploying Your Configuration Data in Command-Line Mode".

The following sections describe how to start and stop the Configuration Deployment Tool.

Note: If the Configuration Deployment Tool is installed on a UNIX/Linux server, the server must have X Windows installed in order to use the ydk.sh script.

5.2.1 Starting the Configuration Deployment Tool (UNIX/Linux)

To start the Configuration Deployment Tool:

1. Go to the <INSTALL_DIR>/database/cdt directory.
2. If your desktop is a Windows machine, and you are connecting to a Unix/Linux machine where the application resides, then you will need an X11 windowing system installed on your Windows machine in order to be able to view the CDT user interface.
3. Change directory to the <INSTALL_DIR>/bin directory
4. Execute the ydk.sh script, enter **./ydk.sh**
   This opens the Sterling Development and Deployment WorkBench.
5. Select Tools > Deployment > Configuration Data Deployment.

This opens the Configuration Deployment Tool Logon window.

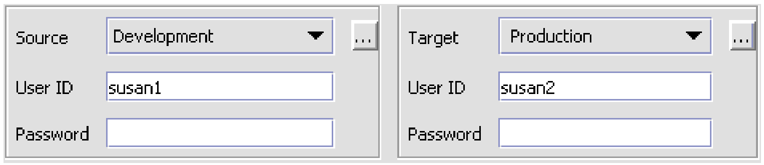
5.2.2 Starting the Configuration Deployment Tool (Windows)

To start the Configuration Deployment Tool:

1. Start the Development and Deployment Workbench. On Microsoft Windows, run the ydk.cmd script from the <INSTALL_DIR>/bin directory.

Note: The Windows console displays WorkBench startup information. Do not close the console while the WorkBench is running. Closing the console closes the tool, and your work is lost.

2. From the Development and Deployment WorkBench menu, choose Tools > Deployment > Configuration Data Deployment. This opens the Configuration Deployment Tool Logon dialog box.



Source: Development
User ID: susan1
Password:

Target: Production
User ID: susan2
Password:

3. Choose the Source button and enter the values appropriate for the source database. Then choose the Target button and enter the values appropriate for the target database.

   When you are finished, close the dialog box. The values you specified are saved automatically and persist from one session to the next.

Note: If you change the name of either source or target database, the transformation settings are lost. To get back your old transformation settings, revert to the old source and target database name.

In the Source database and Target database windows, specify the applicable values as described in Table 5–2.

Table 5–2 Configuration Deployment Tool Logon Dialog Box

Field	Detail
Name	Specify a logical database identifier. For the source, specify the database you want to copy data from. For the target, specify the database to write the data to.
className	Specify the class name of your database driver as follows: • If using Oracle, set to: `oracle.jdbc.OracleDriver` • If using Microsoft SQL Server 2000, set to: `com.microsoft.jdbc.sqlserver.SQLServerDriver` • If using Microsoft SQL Server 2005, set to: `com.microsoft.sqlserver.jdbc.SQLServerDriver` • If using DB2, set to: `com.ibm.db2.jcc.DB2Driver` • If using MySQL, set to: `com.mysql.jdbc.Driver`

Table 5–2 Configuration Deployment Tool Logon Dialog Box

Field	**Detail**
jdbcURL	Specify the URL to connect to the database: • If using Oracle, set to: `jdbc:oracle:thin:@<DatabaseServerHostname/IPaddress><TNSListenerPortNumber>:<DatabaseSID>.` • If using MS SQL Server 2000 or MS SQL Server 2005, set to: `jdbc:microsoft:sqlserver://<Database Server Hostname>:<Port Number>;DatabaseName=<Database name>`. • If using DB2, set to: `jdbc:db2://<Database Server Hostname>:<Port Number>/<Database name>.` • If using mySQL, set to: `jdbc:mysql://<Database Server Hostname>:<Port Number>/<Database name>?useUnicode=true&characterEncoding=UTF-8`.
dbType	Specify the type of database you are running. Enter it in all lower case, as shown: • For Oracle, specify `oracle` • For DB2, specify db2 • For an XML datasource, specify `xml`
folder	If using an XML datasource, specify the complete path of the folder location for the XML files.
httpurl	Only applicable for the target database. Specify a URL for the application server whose data cache is to be refreshed after data is deployed into the target database. Use the syntax: `http://<hostname/ip-address>:<port-number>/application/interop/InteropHttpServlet`, where hostname, IP-address and port-number are the parameters used to connect to the application server.
schema	Specify the schema owner as follows: • If you are using Oracle or DB2 database and the user you specify is different from the schema owner, specify the owner of the schema, otherwise, leave blank. • If you are using SQL Server, leave this blank.
user	Specify the user name associated with the database.

4. In the Logon dialog box, enter the passwords associated with the user names.

   The Deployment Explorer window displays.

To stop the Configuration Deployment Tool:

From the Development and Deployment WorkBench menu, choose File > Exit.

This closes the Configuration Deployment Tool and the Windows console.

6

Transforming Elements Using the CDT

When deploying data from one database instance to another, you can override the values of certain data elements. For example, if your source and target environment network settings (host names, port numbers, and IP addresses) are different, the Configuration Deployment Tool can transform the settings in order to make them appropriate for the target environment.

Transformations are carried out as a pattern match and replace the data in the source database before it is deployed into the target.

Note: The match and replace are carried out for the complete string literal and no wild card search for characters is allowed.

Example Transformation

For example, consider the following configuration XML in the source database:

```
<SubFlowConfig>
  <Link>
    <Properties DeliveryMode=""
     InitialContextFactory="weblogic.jndi.WLInitialContextFactory"
     ProviderURL="t3://localhost:7001" QCFLookup="TEST_AGENT_QCF"
     QName="DefaultAgentQueue" TimeToLive=""/>
   </Link>
</SubFlowConfig>
```

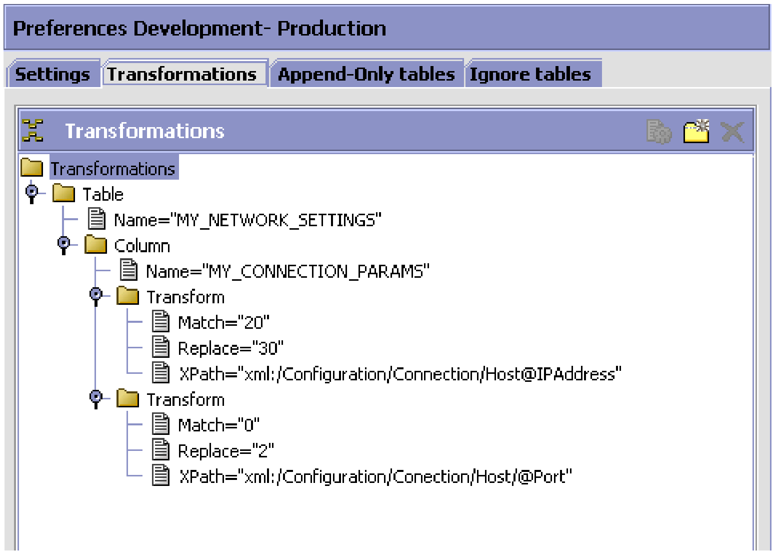
The target database has values for port number as 7221 and QCFLookup as AGENT_QCF, which you do not want overridden by values in the source database. To transform these values, specify the values as described in Table 6–1.

Table 6–1 Transforming Elements

Element	Attribute	Value
ProviderURL		
	Match	Specify 7001 (or 0 to find all occurrences of 0).
	Replace	Specify 7221 (or 2 to replace all occurrences of 2).
	XPath	`xml:/SubFlowConfig/Link/Properties/@ProviderURL`
QCFLookup		
	Match	Specify TEST_ to find all occurrences of TEST_.
	Replace	Leave blank to ensure that TEST_ is removed.
	XPath	`xml:/SubFlowConfig/Link/Properties/@QCFLookup`

Using this example, the Transformation tab would look as shown in Figure 6–1.

Figure 6–1 Transformations Example



```
Transformations
  Table
    Name="MY_NETWORK_SETTINGS"
    Column
      Name="MY_CONNECTION_PARAMS"
      Transform
        Match="20"
        Replace="30"
        XPath="xml:/Configuration/Connection/Host@IPAddress"
      Transform
        Match="0"
        Replace="2"
        XPath="xml:/Configuration/Conection/Host/@Port"
```

To transform elements of the configuration data:

1. From the Deployment Explorer window action bar, choose the *Preferences* icon.

2. In the Preferences window, select the Transformations tab and fill in values, using the information provided in Table 6–1. When you deploy data, these transformation values you specify are deployed along with configuration data.

   Before you deploy data, you must first perform a database comparison as described in Chapter 8, "Comparing Source and Target Databases".

7 Deploying Your Configuration Data

Before deploying configuration data, ensure that you are deploying the correct data by comparing the data and examining any differences.

In addition, ensure that you have addressed rollback issues. The CDT supplies sample backup and rollback scripts. For information on these scripts, see Section 9.4, "Data Rollback Scripts".

Note: Imported Business Processes must be manually re-enabled in the target environment along with any related schedules and adapters.

Also, for the YFS_TRAN_LOCN_ATTRS table, only the configuration data is copied to the target database and the transaction data columns such as Pend_in_volume, Pend_in_weight, and Freeze on variance are not copied to the target database. The weight and volume are recalculated and updated in the target database.

7.1 Deploying Configuration Data

To deploy your configuration data:

1. Compare the two databases (by choosing the *Compare* icon). For this detailed procedure, see Section 8.1, "Comparing Data".
2. From the Comparison Results tree, select the entities you want to deploy.

Tip: When you deploy data for the first time, deploy the entire database. Deploy smaller increments only after you are certain that your source and target databases have relatively few differences.

The available options are as follows:

- For the entire database, choose the correct *Configuration* icon.
- For a specific configuration group, choose the correct *Configuration Group* icon.
- For a specific driver entity, choose the *Driver Entity* icon.

3. From the Comparison Results action bar, choose the *Deploy* icon.
    - If the deployment succeeds, the Status panel displays a success message, the data is committed to your target database, and the cache is updated as specified in the `httpurl` field described in Table 5–2.
    - If the deployment fails, the Status panel indicates the errors to resolve and no data is committed to the target database.
4. If you have deployed data from the YFS_RESOURCE table, restart your application servers in the target environment in order to refresh the cache.

7.1.1 Deploying Your Configuration Data in Command-Line Mode

There may be circumstances under which you want to run or schedule deployment of configuration data without user interface interaction or without viewing the source and target comparison results.

To accomplish this you can deploy your configuration data in command-line mode. When you deploy your data in command-line mode, CDT automatically compares the source and target environments and then deploys the configuration data.

You can configure preferences (such as a reports directory) and parameters that determine the behavior of the comparison operation. When you modify these properties, the changes persist, so you do not need to reset them each time you use CDT. These changes are saved in the `<INSTALL_DIR>/resources/ydkresources/ydkprefs.xml` file.

To deploy configuration data in command-line mode:

1. Start a UNIX/Linux emulator for Windows (use a tool like Cygwin).
2. Open a Telnet window to the UNIX/Linux server the source system is loaded on.
3. In the Telnet window, enter **export DISPLAY=<*ipaddress of pc*>:0.0**
4. Change directory to `<INSTALL_DIR>/bin`
5. Edit the `cdtshell.sh` file to set the properties as follows for a UNIX/Linux server as described in Table 7–1, "Configuration Deployment Tool Properties".
6. Save the file and exit the editor.
7. Deploy data from the command line, enter **./cdtshell.sh**

Table 7–1 Configuration Deployment Tool Properties

Property	Description
JAVA_HOME	Specify the Java installation directory. For example, `JAVA_HOME=D:/JDK_1.5.0_06-b05.`
DB_DRIVER	Specify the path to the database driver. For example, `DB_DRIVER=<INSTALL_DIR>/dbjar/ojdbc14.jar`
DB_EXTN_JAR	Optional. If you have extended any configuration database tables, specify the full path to the `yfsdbextn.jar` file.
SOURCE_DB	Specify the name of the data source as defined in the Configuration Deployment Tool Logon dialog box.
SOURCE_PASSWORD	Optional. If using a database as the source destination for data, specify the password for the database instance.
TARGET_DB	Specify the name of the data target destination for the data. For example, TARGET_DB=xxx (if using a database) or TARGET_DB=xxx (if using an XML file).
TARGET_PASSWORD	Optional. If using a database for the target destination, specify the password for the database instance.

Table 7–1 Configuration Deployment Tool Properties

Property	**Description**
ExportDir	The comparison results are stored in this directory.
ImportDir	This directory should contain the comparison results that are exported. The ImportDir property when present, the source database properties are not used.
DoNotSynchronize	Valid values are Y or N. Y indicates that the comparison results are exported, but are not deployed. N indicates that the comparison results are exported and deployed as well. By default, the comparison results are automatically deployed.

8. Run the `<INSTALL_DIR>/bin/cdtshell.sh` script.

You can also schedule this script to run at any appropriate time.

8

Comparing Source and Target Databases

In order to deploy configuration data into production, you must first compare the two databases and then deploy your changes.

Note: The CDT considers special characters as data when both source and target environment are databases.

8.1 Comparing Data

To compare the source and target databases:

1. From the Deployment Explorer tree, select the data you want to compare.

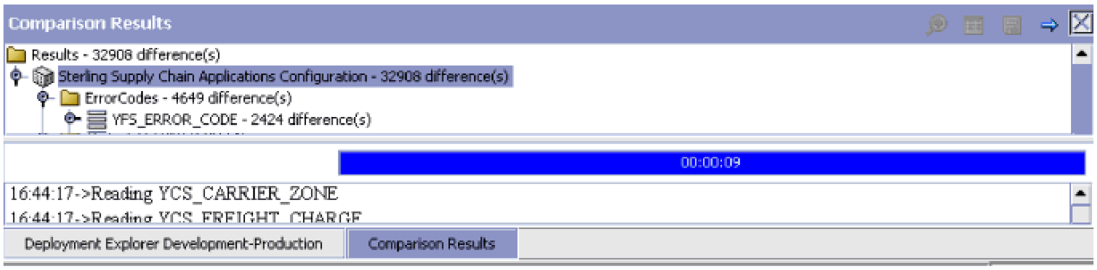
   Tip: When you are deploying data for the ***first*** time, compare the entire database. Compare smaller increments only ***after*** you are certain that your source and target databases have relatively few differences.

   The available comparison groups are as follows:

   - For the entire database, choose the correct *Configuration* icon.
   - For a specific configuration group, choose the *Configuration Group* icon.
   - For a specific driver entity, choose the *Driver Entity* icon.

2. From the Deployment Explorer action bar, choose the *Compare* icon. The Comparison Results window displays on the top right and lists all

differences. The Comparison Results Status panel displays on the bottom right.



Results - 32908 difference(s)
Sterling Supply Chain Applications Configuration - 32908 difference(s)
ErrorCodes - 4649 difference(s)
YFS_ERROR_CODE - 2424 difference(s)

00:00:09

```
16:44:17->Reading YCS_CARRIER_ZONE
16:44:17->Reading YCS_FREIGHT_CHARGE
```

Note: If a table is present in multiple groups or under multiple entities, its difference may be counted multiple times in the total number.

8.2 Examining Database Differences

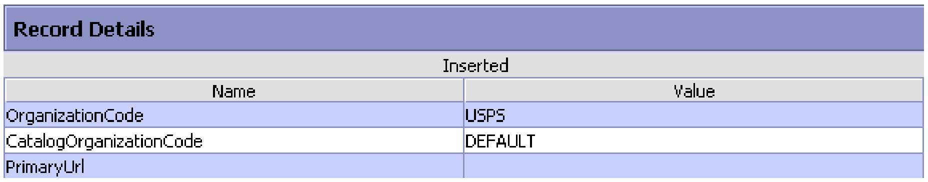
To examine the differences between databases:

1. Choose to compare the two databases.
2. From the Comparison Results tree, expand the corresponding entity and select the table that you want to examine.
3. From the Comparison Results action bar, choose the *Comparison* icon. The range of possible results are as follows:

   The *Unchanged* icon indicates that an entity contains dependent tables that have differences.

   The *Add* icon indicates that a record is inserted to the target database, as shown in Figure 8–1.

Figure 8–1 Record Details Insert Window

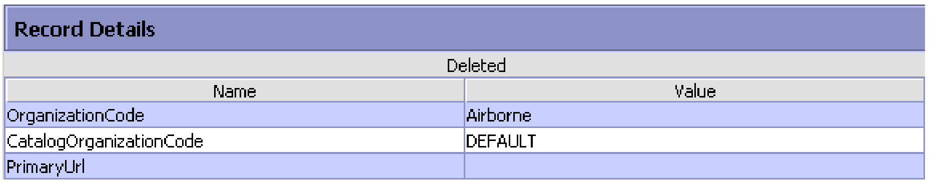


Record Details

Inserted	
Name	Value
OrganizationCode	USPS
CatalogOrganizationCode	DEFAULT
PrimaryUrl	

The *Remove* icon indicates that a record is deleted from the target database, as shown in Figure 8–2.

Figure 8–2 Record Details Delete Window



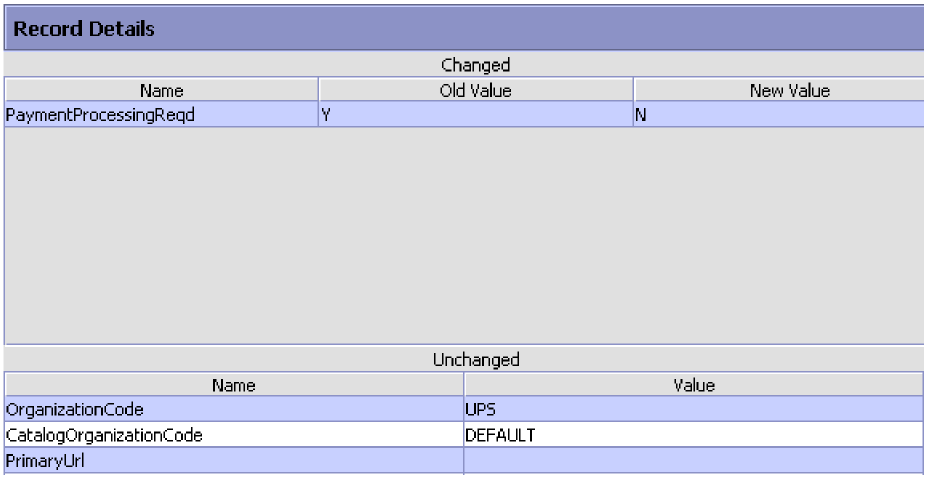
Record Details

Deleted	
Name	Value
OrganizationCode	Airborne
CatalogOrganizationCode	DEFAULT
PrimaryUrl	

The *Modify* icon indicates columns that are updated on the target database as shown in Figure 8–3. It displays information in sections as follows:

- Top section - displays values that are changed on the target database
- Bottom section - displays values that remain the same

Figure 8–3 Record Details Change Window



Record Details

Changed

Name	Old Value	New Value
PaymentProcessingReqd	Y	N

Unchanged

Name	Value
OrganizationCode	UPS
CatalogOrganizationCode	DEFAULT
PrimaryUrl	

4. After examining your data, you may want to generate a report of these differences.

8.3 Exporting Comparison Results

You can export the configuration differences for comparison at a later time or as a backup for your existing configuration.

To export the comparison results into an XML file:

1. Ensure that you have specified a directory location where the comparison report is generated in the Reports Directory field on the Settings tab of your Configuration Deployment Tool Preferences.

2. From the Comparison Results action bar, choose the *Save* icon. From the Windows Explorer, browse to the location specified in the Reports Directory field.

   The Configuration Deployment Tool automatically creates a subdirectory in this directory. For example, if you have specified `D:/reports` in the Reports Directory field and exported the comparison results at 3:40 pm on May 23, 2007, CDT creates the subdirectory as: `D:/reports/export20070523154024`. This new subdirectory contains the `ydkexport.xml` file, which contains the comparison results.

After the comparison completes, you can perform any of the following actions:

- Examine the differences.
- Generate a report of the differences.
- Deploy your changes.

8.4 Generating a Report of Differences

You can generate a report of differences between the source and target databases.

To generate a report of differences:

1. Choose the *Check* icon to ensure that you have specified a `reports` directory.
2. Choose the *Comparison* icon to compare the two databases.
3. In the Comparison Results tree, select the Results node.
4. From the Comparison Results action bar, choose the *Status* icon. The Status panel displays trace messages that enable you to determine the success of the report generation process. The location of the report displays along with a message of the successful creation of reports.
5. From the Windows Explorer, browse to your reports directory. Within the directory you specified, the Configuration Deployment Tool creates a subdirectory named according to the time it was created.

   For example, if you have specified `d:/reports` as the reports directory and generate a report at 3:40 p.m. on May 23, 2007, the CDT creates a subdirectory called `20070523154024` within the `d:/reports` directory.

   This new subdirectory contains the following:

   - An `index.xml` file contains an overall summary of changes as displayed on the UI
   - One XML file for each table that has changes with the details of each change
6. Open the XML files to see the differences.

If you generate another report, a new directory is created and populated with another set of XML files.

8.5 Importing Configuration Differences

You can import configuration differences that are obtained by exporting comparison results.

Note: The Configuration Deployment Tool does not support data that contains special characters when comparing databases, exporting comparison results or importing comparison results.

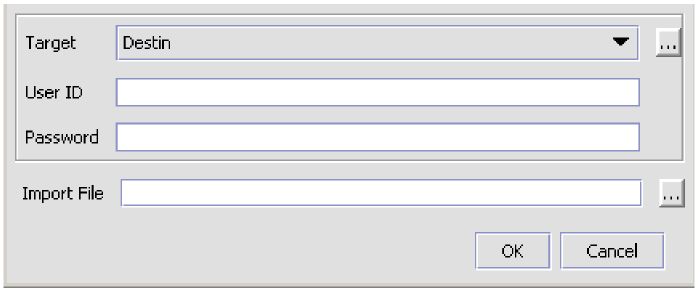
The Import/Export tool loads the differences saved and imports the differences into the target database. No comparison of the differences is done. Therefore, every time you load the exported xml it will show the same data.

Note: The comparison screen shows ALL resources being imported, regardless of whether or not those resources were previously imported into the target database.

For more information about exporting comparison results, see Section 8.3, "Exporting Comparison Results".

To import configuration differences:

1. Run the `<INSTALL_DIR>/bin/ydk.sh` script. This script opens a Microsoft Windows console and starts the Development and Deployment WorkBench.
2. From the Development and Deployment WorkBench menu, choose Tools > Deployment > Import Results. The Configuration Deployment Tool Import dialog box displays.



3. Choose the Target button and enter the values appropriate for the Target database. Then choose the Import File and enter the path of the file to be imported.

   When you are finished, close the dialog box by clicking OK.

4. After the comparison results are loaded, you can perform any of the following actions:

   - Examine any differences, using instructions as specified in Section 8.2, "Examining Database Differences".
   - Generate a report, using the instructions as specified in Section 8.4, "Generating a Report of Differences".
   - Deploy your changes, using instructions as specified in Section 8.5, "Importing Configuration Differences".

9 Troubleshooting

During operations, the Configuration Deployment Tool displays messages in the Status panel that enable you to understand the status of each operation. These messages can be classified as:

- Status
- Warnings
- Unexpected errors

The following section describes various messages that you may encounter and any relevant corrective actions you should take.

9.1 Messages

While using the Configuration Deployment Tool, you may encounter either informational messages or warning messages. These message types are described below.

9.1.1 Informational Messages

Informational messages represent the status of the operation being performed. These messages are displayed in the default color (typically black) in the Status panel. Examples of informational messages include:

- Refreshing database cache
- Deployment operation started
- Reading table YFS_ORGANIZATION

9.1.2 Warning Messages

Warning messages typically require corrective action. They are displayed in red on the Status panel. CDT may produce the warning messages described in this section.

WARNING - FK check failed for table <name> to <name2>

This warning message typically indicates that the configuration data that you are trying to deploy causes inconsistent data in the target database.

To analyze and correct this problem:

1. Determine the size of the data set you are deploying. This error typically occurs when trying to deploy a very small set of data, such as only a driver entity or a configuration group. For example, when deploying a pipeline, this error results if the document type to which the pipeline belongs has not been picked for deployment.

   Try resolving this error by selecting a larger set to deploy. For example, instead of deploying a record, deploy the entire group, if possible.

2. If you still encounter this error for a group or you must only deploy a particular record, try synchronizing the foreign table before deploying the data.

3. Occasionally, inconsistent data in the source database causes this error. If this is the case, you must correct the source of the inconsistency before you proceed.

WARNING - Cache Refresh Failed

This error indicates that CDT was unable to inform the application server cluster on the target environment about the newly deployed configuration changes. The reason the cache refresh failed is displayed on the Status panel.

To analyze and correct this problem:

1. Verify the URL specified in the `httpurl` field for the target database. The `httpurl` is accessible from the Logon dialog box. Ensure that the `httpurl` points to a running instance of the application server and has the following format:

`http://<hostname/ip-address>:<port-number>/yantra/interop/InteropHttpServlet` where hostname, ip-address and port-number are the parameters used to connect to the application server.

2. If your target environment is not running, no action is required. The application automatically reads the latest configuration data when it is started.
3. If the target environment is running, you must manually drop the stale database cache using the System Administration Console. Not performing this step may result in the application not recognizing the changed configuration.

WARNING - The program detected a few abandoned records in the target database.

In most cases, abandoned records are harmless and should not lead to the incorrect operation. By default, CDT leaves them untouched.

This warning typically occurs as a result of the following circumstances:

- When the CDT determines that records do not belong to a valid driver entity (for example, a pipeline for a process type that no longer exists).
- When the CDT has been configured to ignore certain tables without ignoring all dependent tables.

To analyze and correct this problem:

1. Add the `-DShowAbandoned=Y` Java parameter to the `ydk.cmd` script.
2. Run the `ydk.cmd` script. If the CDT finds abandoned records, it dynamically creates a group called "Abandoned Records" and displays them in the Comparison Results window.
3. Examine these records, and then either ignore them or delete them from the target.

9.2 Unexpected Errors

Depending on the severity, messages about unexpected errors are displayed in either of the following places:

- In the CDT Status panel (in red)

- In the Microsoft Windows console used to launch the Configuration Deployment Tool

To analyze and correct these errors:

- If the error indicates an out-of-memory condition, try your previous operation with a smaller set of data.
- Verify that your system specifications comply with the recommendations described in System Requirements.
- You can edit the `-mx` Java parameter in the `ydk.cmd` script to increase the memory available for the Configuration Deployment Tool.

For example, if you were comparing the complete configuration, try comparing one group at a time. The same is true for the deployment operation.

In other cases, the underlying error and detailed trace are displayed. This may point to an incomplete or faulty installation or incorrectly specified runtime parameters.

9.3 Exceptions While Exporting With cdtshell.cmd / sh Scripts

The cdtshell.cmd (or .sh) scripts throw a `java.lang.StringIndexOutOfBoundsException` when exporting configuration data using the CDT with the database as SOURCE_DB and the XML file as TARGET_DB.

To analyze and correct this exception:

Verify that the ExportDir and the folder location of the XML files are not the same location.

Verify that the XML file format is correct.

9.4 Data Rollback Scripts

Before deploying data from a staging to a production environment, it is recommended to take a snapshot of your production configuration data. This snapshot enables you to perform a rollback of the deployment operation in case of failure. The CDT provides the following rollback scripts:

- Backup script - creates multiple files containing data from the configuration data.
- Restore script - uses the files created by the backup scripts to restore the configuration to a previously known good state.

To generate the backup and restore scripts, run the `backupScriptGen.xml` script located in the `<INSTALL_DIR><INSTALL_DIR>/bin` directory. Enter `./ant.sh -f backupScriptGen.xml`.

This script generates sample backup and restore scripts in the `<INSTALL_DIR><INSTALL_DIR>/bin/sample` directory.

Note: The backup files do not represent the entire configuration snapshot (for example, it does not capture the YFS_PERSON_INFO table), so do not use the scripts for deploying data to a different database. Instead, use the CDT to deploy configuration data.

These scripts are only for performing a rollback of configuration data onto the database from which the snapshot was taken.

9.4.1 Customizing the Scripts

You can generate the backup and restore scripts by running the `backupScriptGen.xml` script. This file is located in the `<INSTALL_DIR>/bin<INSTALL_DIR>/bin` directory. You can rename and customize the scripts to suit your business needs. For example, you can modify the script to add your custom configuration tables and modify the path where the data files are stored. These scripts depend on utilities provided by the database vendors.

Note: Running the `backupScriptGen.xml` script creates both backup and restore scripts for Oracle and DB2 databases.

- Oracle scripts depend on `export, import, or sqlplus` utilities. You can modify and use the following scripts:

    - `backup_config_oracle.cmd`
    - `restore_config_oracle.cmd`
- DB2 scripts depend on `export` or `load` utilities. You can modify and use the following scripts:
    - `backup_config_db2.cmd`
    - `restore_config_db2.cmd`

9.4.2 Running the Scripts

Before you deploy any data using the Configuration Deployment Tool, use the backup script to back up your data. These backup data files can then be version controlled.

9.5 Java Error

User receives `java.lang.OutOfMemory` while using CDT.

Modify the ydk.sh file to increase the heap-size to 1024MB.

Follow these steps:

1. Change directory to `<INSTALL_DIR>/bin/`
2. Edit the `ydk.sh` file.
3. Find the following entry: `${JAVA} -DYFS_HOME=D:\install_dir\${HEAP_FLAGS}  -classpath`
4. Change the entry to be:

```
${JAVA} -Xms32m -Xmx1024m
-DYFS_HOME=D:\install_dir\${HEAP_FLAGS} -classpath %CLASSPATH%
```

Index