

SYBASE®

New Features

Sybase Unwired Platform 1.2.2

DOCUMENT ID: DC01092-01-0122-01

LAST REVISED: March 2010

Copyright © 2010 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase trademarks can be viewed at *the Sybase trademarks page*. Sybase and the marks listed are trademarks of Sybase, Inc.® indicates registration in the United States of America.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

New Features 1.2.2	1
New Features	1
Cascade Delete	1
Data Change Notification Interface	2
New APIs	11
Documentation Updates	12
Backup and Restore Overview	12
New Features 1.2.1	19
New Features - Eclipse Edition	19
New Features	21
Documentation Updates	42
Troubleshooting	64
New Features - Visual Studio Edition	69
New Features	70
Documentation Updates	98
Troubleshooting	101
New Features - Sybase Control Center and Unwired Server	103
New Features	106
Enhanced Features	122
Documentation Updates	125
Troubleshooting	133
Index	141

New Features 1.2.2

The Sybase® Unwired Platform 1.2.2 New Features guide includes a description of the new features and a section for documentation updates.

New Features

These new features are included in Sybase® Unwired Platform version 1.2.2.

Cascade Delete

Cascade-delete operations on a parent mobile business object (MBO) automatically cascade changes to the child entity.

Relationships

Cascade-delete behavior is controlled by the types of relationships defined for the MBOs:

- **Composition** – a relationship where the target MBO is a child/detail datatype. A composition relationship can be one-to-one, or one-to-many, but not many-to-one. That is, an MBO can be the child of only one parent MBO.
- **Reference** – a relationship where the target MBO is a master/lookup datatype. A reference relationship is typically many-to-many.

For MBOs that have existing relationships, when an instance of the source (parent) MBO is deleted, all the associated instances (children) of the target MBO are also deleted. This deletion occurs in the consolidated database (CDB) and is reflected when the device is next synchronized. It can also be reflected with the current synchronization if the deletion is caused by a synchronization request. The cascade deletion is recursive in the sense that when an MBO instance is deleted from the CDB as a cascade deletion, that MBO's associations are processed and any composition associations trigger additional cascade deletes of second-level MBOs.

Deployment unit

The relationship element has an additional optional attribute type, with values of either composition or reference, as shown:

```
<association name="items" mobile_object_name="order_item" type="composition"
x_to_many="true">
```

```
<association name="product" mobile_object_name="product" type="reference"
x_to_many="true">
```

Admin API

To specify the relationship type, use the HTTP API with a URL in this format:

```
http://host:port/onepage/servlet/UWPServlet?  
app=uep&cmd=MobileObjectAdmin.setAssociationType&package_name=packa  
ge_name&mobile_object_name=mbo_name&association_name=assoc_name&  
association_type=type&authenticate.user=username&authenticate.passw  
ord=password
```

Deployment default

During redeployment, if the deployment unit does not specify the relationship type and the relationship already exists in the CDB, then its previous known relationship type is applied. This means you need not set the relationship type after every deployment.

Data Change Notification Interface

Data change notification (DCN) provides an HTTP interface by which enterprise information system (EIS) changes can immediately be propagated to Unwired Server.

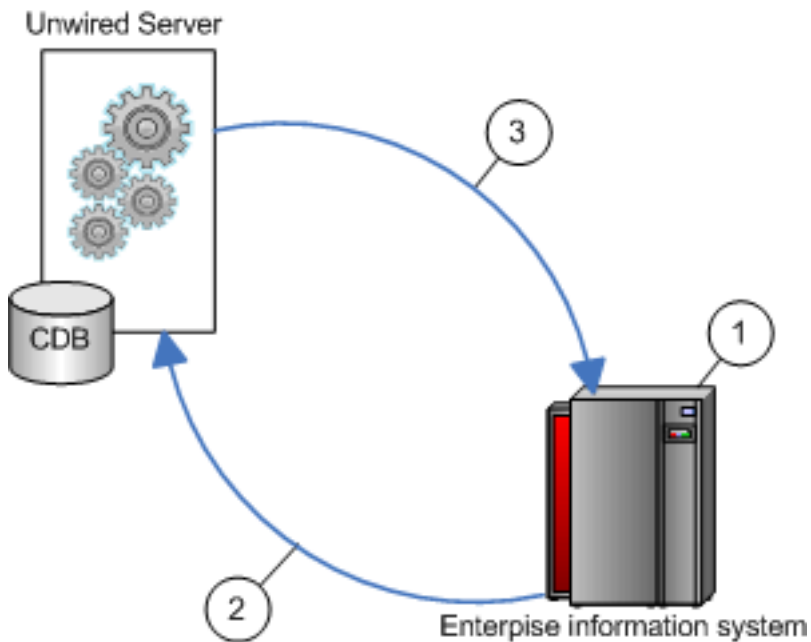
All DCN commands support both GET and POST methods. The EIS developer creates and sends a DCN to Unwired Server through HTTP GET or POST operations. The portion of the DCN command parameters that come after `http://host:port/onepage/servlet/UWPServlet?` can all be in POST, or any *var=name* can be in either the URL (GET) or in the POST. The **authenticate.password** parameter is an especially good candidate for including in the POST method, as well as any sensitive data provided for attributes and parameters because the HTTP POST method is more secure than HTTP GET methods.

Note: Enter the HTTP request on a single line.

You can use DCN with or without payload:

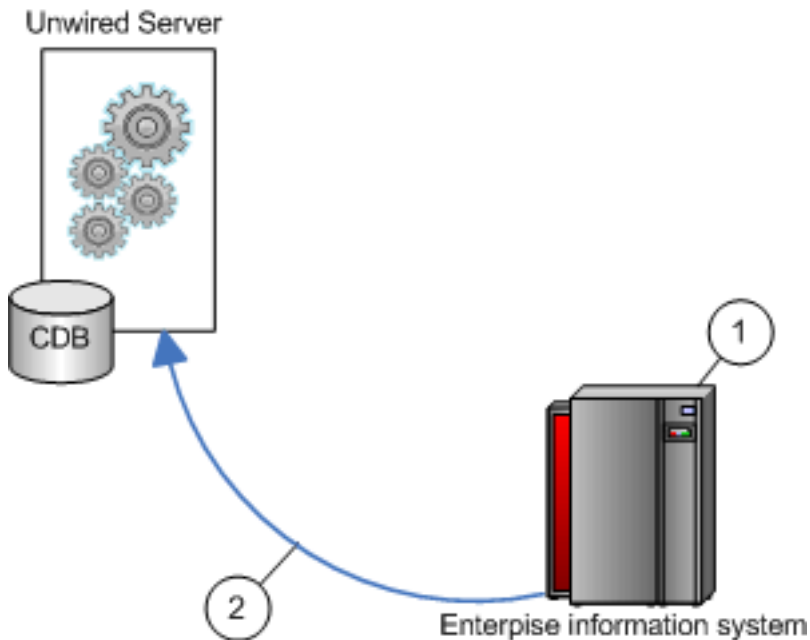
1. To instruct Unwired Server to invoke an operation without payload, the DCN instructs Unwired Server to do something, based on the Unwired Server configuration or MBO settings as shown here.

Figure 1: DCN without payload



- (1) EIS data changes, which triggers the DCN.
 - (2) The DCN issues instructions to Unwired Server.
 - (3) Unwired Server performs the DCN task, for example, invalidating and refreshing an MBO.
2. In the method shown here, the Unwired Server cache is updated directly. The actual data (payload) is applied to the cache, through either an **:upsert** (update or insert) or a **:delete** operation.

Figure 2: DCN with payload



- (1) EIS data changes, which triggers the DCN.
- (2) The DCN applies data changes, for example deleting a data row, directly to the cache (CDB).

Be familiar with the EIS data source from which the DCN is issued. DCNs can be created and sent based on:

- Database triggers
- EIS system events
- External integration processes

Invoking Operations Through Data Change Notification

Data change notifications (DCNs) can instruct Unwired Server to invoke a mobile business object (MBO) operation with a set of specified parameters. Changes to the Unwired Server cache are dependent on the operation's specified behavior, which was defined during MBO development.

These examples show DCN without payload.

Syntax

DCNs that invoke operations require a JavaScript Object Notation (JSON) string (dcn_request) that contains information about the MBO operation:

```
http://unwired_server.sybase.com:unwired_server_port/onepage/  
servlet/UWPServlet?
```



```
&cmd=dcn
&username=userName
&password=password
&package=unwired_server_PackageName
&dcn_request={ "pkg": "TestPackage",
  "messages": [ { "id": "1", "mbo": "mboName", "op": "operationName",
    "cols": { "operationParameters" } } ] }
&dcn_filter=fully_qualified_name_of_dcn_filter
```

Parameters

- **unwired_server** – Unwired Server host name to which the DCN is issued.
- **unwired_server_port** – Unwired Server port number
- **&username** – authorized Unwired Server user with permission to modify the MBO.
- **&password** – authorized user's password.
- **&package** – Unwired Server package that contains the MBO. The format is package:version. For example, e2e_package:1.0.
- **&dcn_request** – the JSON string that contains operation name and parameters, which must include:
 - Package name (pkg)
 - A list of messages (messages), where each message includes:
 - A message ID (id) used to report back the status.
 - Mobile business object name (mbo).
 - Operation name (op): an operation name of the specified MBO, or an alternate read operation.
 - Bindings (cols): operation parameter names and values covering at least all primary key attributes of the MBO.
- **(Optional) &dcn_filter** – custom filter used to convert any JSON strings. By default, Unwired Server expects all DCN requests to be a valid JSON string. Use the DCN filter to convert incoming DCN request strings to a valid JSON string.

Examples

- **MBO operation invocation** – This JSON string (included in a DCN request) invokes the Department MBO's UpdateDepartment operation to update the department ID, Name, and HeadID:

```
{ "pkg": "TestPackage",
  "messages": [ { "id": "1", "mbo": "Department", "op":
    "UpdateDepartment",
    "cols": { "UpdateDepartment_department_DepartmentID": "3333",
      "UpdateDepartment_department_DepartmentName": "My
    Department",
      "UpdateDepartment_department_DepartmentHeadID": "501" } } ] }
```

- **Multiple operations in one DCN** – This example includes multiple messages within one DCN included in the JSON message array:

```
dcn_request={ "pkg": "TestPackage" ,
  "messages": [
    { "id": "1" , "mbo": "Department" , "op": "UpdateDepartment" ,
      "cols": { "UpdateDepartment_department_DepartmentID": "3333" ,
        "UpdateDepartment_department_DepartmentName": "My
        Department" ,
        "UpdateDepartment_department_DepartmentHeadID": "501" } }
    { "id": "2" , "mbo": "AnotherMBO" , "op": ":upsert" ,
      "cols": { "AnotherMBOID": "99" ,
        "MyColumn": "Test ValueMy" } } ] }
```

Usage

A DCN that invokes an operation follows these rules:

- All parameters used in the MBO operation must be included.
- All columns used in the operation must use the MBO parameter name.
- Any explicit value specified in the DCN request takes precedence over a personalized value. To use a personalized value in DCN processing, do not include the personalized parameter name and its value in the DCN request. The personalized value is then inferred by Unwired Server and added to the supplied bindings. Similarly, omit parameters with default values to use previously specified default values.

Modifying Data Using Data Change Notification

Data change notifications (DCNs) with payload update Unwired Server cache directly, through either an **:upsert** (update or insert) or a **:delete** operation.

Syntax

DCN with payload requires a JavaScript Object Notation (JSON) string (`dcn_request`) that contains one or more **:upsert** and/or **:delete** operations that are applied to the Unwired Server cache (CDB).

```
http://unwired_server.sybase.com:unwired_server_port/onepage/
servlet/UWPServlet?
&cmd=dcn
&username=username
&password=password
&package=unwired_server_PackageName
&dcn_request={ "pkg": "dummy" , "messages":
  [ { "id": "1" , "mbo": "CustomerWithParam" , "op": ":upsert" , "cols":
    { "id": "10001" , "fname": "Adam" } } ] }
&dcn_filter=fully_qualified_name_of_dcn_filter
```

Parameters

- **unwired_server** – Unwired Server host name to which the DCN is issued.
- **unwired_server_port** – Unwired Server port number.
- **&username** – authorized Unwired Server user with permission to modify the MBO.
- **&password** – authorized user's password.
- **&package** – Unwired Server package that contains the MBO. The format is package:version. For example, e2e_package:1.0.
- **&dcn_request** – the JSON string that contains the data, which must include:
 - Package name (pkg)
 - A list of messages (messages), where each message includes:
 - A message ID (id) used to report back the status.
 - Mobile business object name (mbo).
 - Operation name (op): ":upsert", ":delete", or an operation name of the specified MBO. ":upsert" and ":delete" are special operation names to specify payload data.
 - Bindings (cols): a list of name/value pairs covering at least all primary key attributes of the MBO. For payload data, the bindings contain the payload specified as MBO attribute/parameter names and their values.
- **(Optional) &dcn_filter** – custom filter used to convert any JSON strings. By default, Unwired Server expects all DCN requests to be a valid JSON string. Use the DCN filter to convert incoming DCN request strings to a valid JSON string.

Examples

- **Upsert example** – This JSON string included in a DCN contains a single :upsert operation that updates or inserts (upserts) data in the Unwired Server cache for the Department MBO.

```
dcn_request={ "pkg" : "TestPackage" ,
"messages" :
  [ { "id" : "1" , "mbo" : "Department" ,
      "op" : ":upsert" ,
      "cols" : { "DepartmentID" : "3333" ,
                "DepartmentName" : "Test Value" ,
                "DepartmentHeadID" : "501" } } ]
}
```

- **Delete example** – Example JSON string included in the DCN sent to Unwired Server used to delete a row of data from the Unwired Server cache for the Department MBO:

```
dcn_request={ "pkg" : "TestPackage" ,
"messages" : [ { "id" : "1" , "mbo" : "Department" ,
"op" : ":delete" ,
"cols" : { "DepartmentID" : "3333" } } ] }
```

Usage

Follow these guidelines when constructing a DCN:

- The format of non string data is the same as parameter default values in Unwired Workspace. For example, specify timestamp values in a format similar to 2009-03-04T17:03:00+05:30.
- Base binding names on MBO names rather than EIS names.
- The :upsert operation requires:
 - All MBO Primary key attributes to properly upsert the data.
 - Any other MBO attributes used in the upsert.
 - All columns used in the operation must use attribute names (not column names to which it is mapped).
- The :delete operation requires:
 - The MBO Primary key attribute to properly delete the data.
 - That all columns used in the operation must use attribute names (not the column names to which it is mapped).

Data Change Notification Results

Each binding in a data change notification (DCN) request is associated with an ID. The result status of the DCN request is returned in JavaScript Object Notation (JSON) format, and includes a list of IDs followed by a Boolean success field and status message, in case of error.

In response to payload and MBO operation DCNs, Unwired Server sends the requestor a JSON string containing details about the success and or failure of the operations. This example shows the JSON format for a DCN result for a request of three IDs (recID1, recID2, recID3). The example has been formatted using new lines, and indentations, which are not present in an actual response:

```
[
  {
    "recordIDs" :
    [
      "recID1" ,
      "recID2"
    ],
    "success":true,
    "statusMessage": " "
  },
  {
    "recordIDs" :
    [
      "recID3"
    ],
    "success":false,
    "statusMessage": "bad msg2"
  }
]
```

This example is unformatted:

- Successful operation:

```
[{"recordIDs":["1"],"success":true,"statusMessage":""}]
```

- Failed operation using tildas instead of colons:

```
[{"recordIDs"~["1"],
  "success"~false,"statusMessage"~"Error inferring attribute
bindings from EIS bindings {DepartmentID\u003d10000,
  DepartmentAlias\u003dTest,
  DepartmentHeadID\u003d501}"}]
```

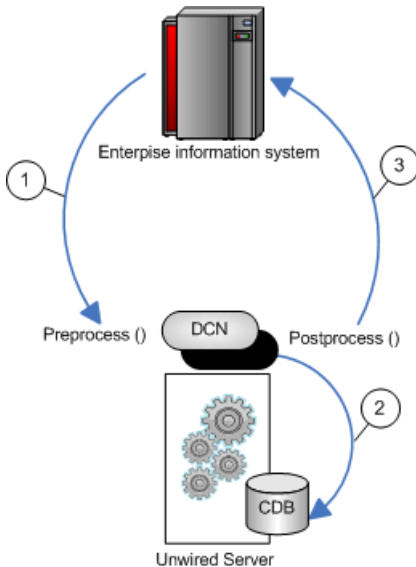
Data Change Notification Filters

Data change notification (DCN) requests need not always be in the format Unwired Server expects. You can deploy a DCN filter to Unwired Server and reference it in the DCN request. Unwired Server allows the filter to preprocess the submitted DCN. The filter converts raw data in the DCN request to the required JavaScript Object Notation (JSON) format. The filter can also postprocess the JSON response returned by the Unwired Server into the format preferred by the back end (which is governed by the implementation in the filter class).

The filter interface `DCNFilter` is located in the `com.sybase.sup.dcn` package in the `onepage.jar` file. All classes that implement a DCN filter should implement this interface. The functions available in the interface are:

- **String preprocess(String blobDCNRequest, Map<String, String> requestHeaders);**
– takes the DCN request as a binary large object (BLOB), converts it into a valid JSON DCN request format, and returns the same.
- **String postprocess(String jsonDCNResult, Map<String, String> responseHeaders);**
– takes the DCN result in a valid JSON format, converts it to the EIS-specific format, and returns the same.

Figure 3: DCN filter flow



1. Changed data is sent from the EIS to Unwired Server via a DCN request, where any preprocessing of the data occurs. For example, the EIS data could be sent to Unwired Server as XML where the preprocess filter converts the data to JSON.
2. The DCN executes. For example, apply data changes directly to the Unwired Server cache.
3. Postprocessed DCN response is sent to the originating EIS as an HTTP response to the original DCN request. For example, the JSON response is converted to XML.

Implementing a Data Change Notification Filter

Write and deploy preprocess and postprocess DCN filters to Unwired Server.

When specifying filters, add a `dcn_filter` parameter to the base URL, and to the parameters specified in the DCN request section. The `dcn_filter` parameter specifies the fully qualified name of the filter class, which must be in a valid CLASSPATH location so Unwired Server can locate it using its fully qualified name. If the filter class is in an Unwired Platform root folder, it is automatically propagated across a cluster.

JSON requires colons to define the object structure, but since colons have a special function in HTTP URLs, use the tilde character "~" instead of colons ":" when implementing the DCN filter, so the JSON `dcn_request` string can be passed as an HTTP GET or POST parameter:

```
dcn_request={ "pkg"~"TestPackage" ,
  "messages"~[ { "id"~"1" , "mbo"~"Department" , "op"~"~upsert" ,
  "cols"~{ "DepartmentID"~"3333" ,
    "DepartmentName"~"My Department" ,
    "DepartmentHeadID"~"501" } } ] }
```

The `dcn_request` is in a format that is specific to the back end. The filter class can preprocess to the JSON format expected by Unwired Server.

1. Write the filter. For example:

```
import java.util.map;
import com.onepage.fw.uwp.shared.uwp.UWPLogger;
import com.sybase.sup.dcn.DCNFilter;

public class CustomDCNFilter implements DCNFilter
{
    String preprocess(String blobDCNRequest, Map<String,String>
headers) {
        String result = blobDCNRequest.replace('~', ':');
        return result;
    }

    String postprocess(String jsonDCNResult, Map<String,String>
responseHeaders) {
        String result = jsonDCNResult.replace(':', '~');
        return result;
    }

    public static void main( String[] args ) { }
}
```

2. Shut down Unwired Server and deploy the DCN filter:

- a) Package your DCN filter class in a JAR file.
- b) Copy the JAR file to Sybase
 - \UnwiredPlatform_installation_directory\Servers
 - \UnwiredServer\lib.

Note: The JAR file will not be propagated automatically across cluster nodes. You must stop the node to which you want to propagate the JAR file, then copy the JAR file manually.

3. Restart Unwired Server.

New APIs

Several new device client APIs in Sybase Unwired Platform version 1.2.2 enhance performance.

See the *Sybase Unwired Platform 1.2.2 Client Object API Cookbook* at <http://www.sybase.com/developer/library/suptechcorner> for information about these APIs, which include:

- `Evict` – removes orphan mobile business object instances.
- `IsPendingOverwritten` – checks to see if the record to be updated is overwritten after the sync.
- `<MBO>PendingState` – returns the updated and saved state of the object.

New Features 1.2.2

- `FindByOperationId` – helps identify which record each error message in the sync log refers to when changes made on the client side are uploaded to the server.
- `GetDataCount(Query query)` – public static int called by the client to learn in advance the number of mobile business objects this query will fetch if it is passed to the `Find(Query)` method.
- `Load[relationship_name](List<parent_mbo_name>)` – this batch preload API improves performance by batching and preloading all relationship data in mobile business objects into memory so that multiple calls to the database for relationship data are not necessary.

Documentation Updates

These documentation updates apply to Sybase Unwired Platform version 1.2.2.

Document	Update
Sybase Unwired Platform 1.2.1 > Sybase Unwired Platform 1.2.1 New Features > New Features - Eclipse Edition > Documentation Updates > Preparing the Unwired Platform Environment for SAP Connections	Step 4: For the component Eclipse Unwired Workspace, copy <code>librfc32.dll</code> and <code>sapjcoRFC.dll</code> into the following target directories: <ul style="list-style-type: none">• <code>C:\WINDOWS\system32</code>• <code><SUP Installation root>\UnwiredPlatform-1_2\JDK1.6.0_12\bin</code>
<i>Backup and Restore Overview</i> on page 12	The Backup and Restore topics provide information for backing up and restoring the installation file system, consolidated database, and transaction logs for SUP 1.2.2.

Backup and Restore Overview

Learn how to plan Sybase Unwired Platform backup schedules to support disaster recovery planning. Provides information for backing up and restoring the installation file system, consolidated database, and transaction logs; and related information for mobile device client users.

Backup of the Installation File System

Make sure to create regular and complete backups of the Sybase Unwired Platform installation files and directories, which are modified as part of regular operation and with configuration changes.

Examples of these Sybase Unwired Platform installation files and directories include:

- `<SUP_HOME>\Servers\UnwiredServer\Repository`

- <SUP_HOME>\Servers\UnwiredServer\config
- <SUP_HOME>\Servers\UnwiredServer\bin
- <SUP_HOME>\ Servers\UnwiredServer\SQLAnywhere11
- <SUP_HOME>\ Servers\UnwiredServer\logs
- <SUP_HOME>\Servers\UnwiredServer\tomcat\webapps\onepage
 \config
- <SUP_HOME>\Servers\UnwiredServer\tomcat\conf

Additionally, batch scripts (for example, <SUP_HOME>\Servers\UnwiredServer\bin) may be automatically modified by the Sybase Unwired Platform installer, or manually modified later by a Sybase Unwired Platform administrative user. Make sure these files are also backed up.

Instead of backing up these individual artifacts, Sybase recommends that you perform regular backups of the entire Sybase Unwired Platform installation folder. Ideally, include the Sybase Unwired Platform installation directory in your disk backup schedule. At the same time the folder and disk backup is performed, update the Windows registry so it matches the state of the backup.

Plan the frequency of the file system backups to coincide with any changes made to the system, including metadata changes (such as deployment of new Mobile Business Object packages to the server), or configuration changes (such as new enterprise information system connection). To maintain a consistent backup state, Sybase recommends you back up the consolidated database at these times as well.

Backup of the Consolidated Database

Create regular and complete backups of the Sybase Unwired Platform consolidated database (CDB) and its transaction logs. These instructions assume you are using SQL Anywhere as your Sybase Unwired Platform database server.

SQL Anywhere provides backup and recovery tools. Determine your tolerance for data loss, and your expectations for recovery time, then design a backup and recovery plan for your specific enterprise needs.

Following are basic instructions for making your databases recoverable in case of a disk crash or catastrophic computer failure. If you require more comprehensive recovery policies, Sybase offers professional services specifically related to this topic.

The Sybase Unwired Platform consolidated database contains the metadata of deployed applications, and transient and cached data that is sent to mobile devices.

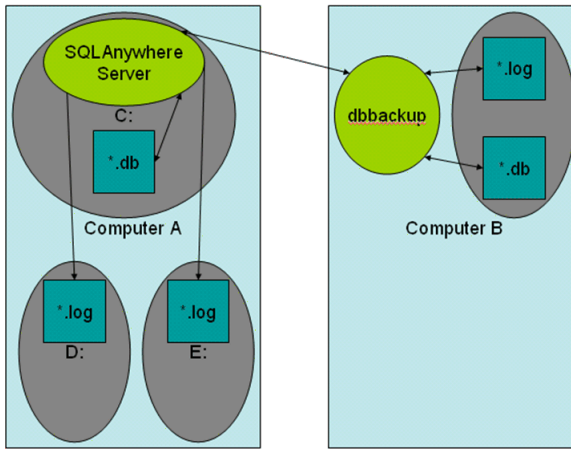
Frequent backups of the consolidated database are required to maintain the deployed applications metadata. The backups may also help restore the transient and cached data (to a large degree) in case of crashes or corruption, and may help mobile device clients from having to perform a full refresh.

Sample Backup and Recovery Plan

Provides a basic backup and recovery plan.

This diagram shows the architecture for a reasonably reliable backup and recovery strategy. Only Sybase Unwired Platform components related to database recovery are included.

Figure 4: Sample backup and recovery plan



Shown in the diagram:

- Computer A is where the SQL Anywhere database server is installed and runs under Sybase Unwired Platform. There are three physical disks on this computer:
 - The C: drive has the SQL Anywhere server and the database files (uam1 .db, clusterdb .db), which hold critical data that Sybase Unwired Platform requires to function.
 - The D: and E: drives hold identical copies of the transaction log files (uam1 .log, and clusterdb .log). Using SQL Anywhere terminology, the D: drive holds the regular transaction log, and the E: drive holds the mirrored transaction log.
- Computer B is for long term backup, and requires only one drive (or backup tapes). Run the **dbbackup** utility from this computer periodically to obtain a full backup of the * .db and * .log files from Computer A.

Failure and Recovery Scenarios

Describes several failure scenarios (using the Sample Backup and Recovery Plan setup), how recovery works, and implications for Sybase Unwired Platform operation.

Disk C has an unrecoverable error. The *.db files have been lost.

Recovery: Install a replacement disk, and use standard file restore procedures to reinstall the SQL Anywhere software, and whatever else is needed. If the restore returns the * .db files,

there is no harm, but do not rely on these files to be valid. Instead, copy the last backup version of the *.db files across from Computer B.

Next, start the SQL Anywhere server, which detects that the *.db files are not up-to-date with the checkpoints in the *.log files on drives D: and E: (which are unaffected by the C: drive failure). The server automatically replays transactions recorded in the transaction log to bring the database back to the state of all committed transactions at the time of the C: drive failure.

Sybase Unwired Platform can then start up and run normally. Sybase Unwired Platform mobile device clients are not affected except by the inability to sync between the time of the failure, and the time at which the recovery process has completed.

Disk D: or E: failure. One of the *.log files has been lost.

Recovery: Install a replacement disk and restore from backups.

Once the disk has been restored, copy the *.log files from the drive that did not fail to the one that failed. Restart the failed drive.

Complete failure of Computer A, and disks lost.

Recovery: See Restore the Consolidate Database for instructions. This should be a very infrequent event.

In this scenario, the database has lost all transactions since the previous backup to Computer B. Any Sybase Unwired Platform mobile device clients that synchronized between the time of the previous backup and the time of the failure cannot now sync. Clients must delete their UltraLite database and start fresh. Any pending operations on these clients are lost. Clients that have not synchronized since the previous backup are unaffected.

Configuring Your Databases for Transaction Logs on Separate Drives

Use SQL Anywhere tools to configure and manage your consolidated databases for transaction logs and mirrored transaction logs on separate drives.

These tools are located in the <SUP_HOME>\Servers\UnwiredServer\SQLAnywhere11\BIN32 directory. Add this directory to your PATH.

1. Shut down the SQL Anywhere server.
2. From the SQLAnywhere11 directory, run:

```
dblog -t D:\logs\uaml.log -m E:\logs\uaml.log uaml.db
```

Output from a successful run of this command will look similar to:

```
SQL Anywhere Transaction Log Utility Version 11.0.1.2250
"uaml.db" was using log file "uaml.log"
"uaml.db" was using no log mirror file
"uaml.db" is now using log file "D:\logs\uaml.log"
"uaml.db" is now using log mirror file "E:\logs\uaml.log"
```

```
Transaction log starting offset is 0000746294  
Transaction log current relative offset is 0001015441
```

3. Repeat this command for the `clusterdb` database:

```
dblog -t D:\logs\clusterdb.log -m E:\logs\clusterdb.log  
clusterdb.db
```

4. Restart SQL Anywhere server to start using these logs on other disks.

Performing a Remote Backup

Perform a remote backup using the sample recovery plan as an example.

Once your database is properly mirrored with transaction logs on separate disks, perform an initial backup of the consolidated database. Use the tools in the `<SUP_HOME>\Servers\UnwiredServer\SQLAnywhere11\BIN32` directory. Add this directory to your `PATH`.

Note: In some configurations, the **dbvalid** and **dbbackup** commands do not work remotely.

1. Use the **dbvalid** utility to validate the integrity of the database. The validation must be performed when there are no active connections to the server. Use the **dblocate** utility to locate the actual name of the database server in a particular installation. Typical names take the form of `<clustername>_primary`. The `clusterdb` is present if Unwired Server has been configured to run as a cluster.

```
dbvalid.exe -c "DBF=uaml.db;UID=dba;PWD=SQL"
```

```
dbvalid.exe -c "DBF=clusterdb.db;UID=dba;PWD=SQL"
```

2. On Computer B, verify that SQL Anywhere software is installed, and the `PATH` is set. If you are running Sybase Unwired Platform as a cluster, you already have that software installed under the Sybase Unwired Platform installation directory. Otherwise, install another copy of Sybase Unwired Platform there. You need not add this installation to your cluster; you can even use a Developer Edition installation if you like.
3. Once Computer B is set up with the SQL Anywhere tools, run:

```
dbbackup -c "ENG=<clusterName>_primary;DBN=uaml;UID=dba;PWD=SQL"  
\SQLAnybackup
```

```
dbbackup -c  
"ENG=<clusterName>_primary;DBN=clusterdb;UID=dba;PWD=SQL"  
\SQLAnybackup
```

This creates `uaml.db`, `uaml.log`, `clusterdb.db`, and `clusterdb.log` in the `\SQLAnybackup` directory on Computer B.

4. As a precaution, validate the backups are suitable for recovery:
 - a) On Computer B, create a temporary working directory (such as `\tmp`).
 - b) Under the temporary directory, create an identical directory structure for the two log locations. You may need to use the **subst** command to map local directories to drive letters used on Computer A.

- c) Copy *.log to these locations.
- d) Run **dbvalid** on the \tmp copy of the .db file.
WARNING: Do not run **dbvalid** on the backup copy itself (in the \SQLAnybackup directory of this example). The command runs, but corrupts your .db file so it cannot be used in recovery.
- e) If validation succeeds, you are assured that your backup in \SQLAnybackup can be used for recovery. You can delete the files in the \tmp and log directories.
 If validation fails, the backup is not usable for recovery and you should try again.

Next

Sybase makes these recommendations for setting up your backup schedule:

- Schedule backups from Computer B for some reasonable frequency so that, in the worst-case failure of a complete failure of Computer A, you do not lose too many transactions.
- Perform adhoc backups when there has been some major deployment of new Mobile Business Object packages to Sybase Unwired Platform. Otherwise you may lose those packages and have to redeploy them.

Restoration of the Installation File System

Restore the Sybase Unwired Platform installation file system from a backup.

To perform a normal restoration, use the file or disk backup utilities used to perform the backup. Sybase recommends that you save the current installation directory before you restore from backup.

Note: You may also need to restore the Windows registry from the backup done at the same time.

Restoration of the Consolidated Database

Restore the Sybase Unwired Platform consolidated database and transaction logs from backup after complete failure.

As discussed in *Failure and Recovery Scenarios*, if only one of C:, D:, or E: drives fail, recovery should be automatic once you have completed the appropriate tasks.

These steps are required in case of complete failure of all three drives:

1. Restore the computer's C:, D:, and E: drives from backup.
2. Delete the *.db, *.log files from their normal places after you have restored the file system.
3. Copy the *.db and *.log files from Computer B's backup directory to the normal locations on Computer A.

Note: Copy the *.log files twice—once to the normal transaction logs directory, and once to the mirrored transaction logs directory.

4. Restart Sybase Unwired Platform.
5. If there have been package deployments or other cluster-affecting operations since the last database backups, the file-system data corresponding to the packages may be out-of-sync with the database contents related to these packages. If this has occurred, the Sybase Unwired Platform servers cannot fully start. The `m1.log` file indicates a mismatch between the local cluster version and that of the current cluster. To recover from this:
 - a. Choose one of the Unwired Servers.
 - b. Shut down that server.
 - c. Edit the `sup.properties` file (`UnwiredServer` directory), and change the `cluster.version` property value to match that of the current cluster as reported in the logs.
 - d. Run `updateProps -r` from the same directory to apply the change into the `clusterdb`.
 - e. Restart that Unwired Server.

Note: This server should be able to take over as the Sybase Unwired Platform primary. Sybase recommends you redeploy all your packages (using the **UPDATE** option) to make sure all the packages you expect to be available really are. All of your Sybase Unwired Platform clients should delete their UltraLite databases, and perform full synchronizations.

New Features 1.2.1

This document includes the Sybase Unwired Platform new features for 1.2.1, for both the Eclipse and Visual Studio editions.

New Features - Eclipse Edition

New and enhanced features, updated documentation, and new troubleshooting topics.

The tables below provide a brief description of each feature and links to associated topics.

New Sybase Unwired Platform features

Feature	Topics
A multilevel (chained) insert operation allows Web service insert operations between mobile business objects that have a defined relationship during one synchronization.	<i>Creating Multilevel Insert Operations for Web-Service Mobile Business Objects</i> on page 22
Optimize device application and Unwired Server performance by including a cache update policy when developing mobile business objects.	<i>Cache Update Policy</i> on page 25
Specify custom result checking logic for SAP mobile business objects.	<i>Adding an SAP Result Checker</i> on page 35
Import Visual Studio projects into Unwired WorkSpace (Eclipse).	<i>Importing Visual Studio Projects into Eclipse</i> on page 41
Configure an SAP operation, so if it is successful, it always commits.	<i>Configuring an SAP Operation to Commit</i> on page 41
Use enterprise information system (EIS) properties as mobile business object (MBO) parameters when making direct connections from an SAP MBO to an EIS.	<i>Modifying SAP Direct Connection Properties</i> on page 42

Documentation updates

Description	Topics
Updated playback and synchronization parameter documentation.	<i>Playback and Synchronization Parameters</i> on page 43

New Features 1.2.1

Description	Topics
Corrected formatting to display entire path for the location of the driver.	<i>Configuring Your Environment to Use a JDBC Driver</i> on page 47
Updated to reflect the automatic screen creation feature.	<i>Using Drag and Drop to Add Mobile Business Objects to the Flow Design</i> on page 48
Updated to correct the description of how the Delete Operation Screen option in Device Application Designer preferences works.	<i>Automatic Screen Creation</i> on page 50
Added the new Save Context and PIM actions.	<i>Screen Design Palette Options</i> on page 50
Added the Logical Type and Data Type properties.	<i>Choice Properties</i> on page 54 <i>Radio Group Properties</i> on page 55
Added the Launch PIM Application option.	<i>Adding a BlackBerry PIM Action</i> on page 56
Removed the note about linked parameters not being supported.	<i>Generating a Windows Mobile Device Application</i> on page 58
Replaced "Configuring Your Environment for SAP" with "Preparing the Unwired Platform Environment for SAP Connections."	<i>Preparing the Unwired Platform Environment for SAP Connections</i> on page 45
Added new topic for developing a device application for the Win32 .NET platform.	<i>Developing Device Applications for Win32 .NET Platforms</i> on page 61
Corrected the implementation class names that are documented in "Writing a Custom Result Set Filter."	<i>Correction to Writing a Custom Result Set Filter</i> on page 64

Troubleshooting

Feature	Topics
Configure Unwired Server properties only from the Administration Console or by editing the sup.properties file.	<i>Restrictions for Configuring Unwired Server Properties</i> on page 64
Mobile business object (MBO) argument and column names have no character length limits.	<i>Argument and Column Name Length Limitations</i> on page 65
If you cannot preview mobile business objects that use a JDBC driver, set <code>automommitPreviewTransaction</code> to true .	<i>Cannot Preview Data</i> on page 65

Feature	Topics
A Web service that contains a parent and a child hierarchical data structure with the same column names cannot be deployed.	<i>Troubleshooting Mobile Business Object Web Service Deployment Failure</i> on page 65
In some cases, the default value supplied by the device application is not recognized as a sync value in Unwired Server.	<i>Default Values are not Recognized as Synchronization Values</i> on page 66
SQLE_TOO_MANY_PUBLICATIONS error received when synchronizing mobile business objects.	<i>SQLE_TOO_MANY_PUBLICATIONS Error</i> on page 66
Device Application Designer does not generate GUI fields for enterprise information system connection property parameters so mobile users cannot configure connection properties value in runtime.	<i>Device Application Designer Does Not Generate GUI Fields</i> on page 66
Trailing space causes synchronization failure.	<i>Trailing Space Causes Synchronization Failure</i> on page 67
BlackBerry devices encounter out of memory errors if trying to display more than 6000 records (rows) of data.	<i>BlackBerry Devices Display a Maximum of 6000 Records</i> on page 67
When trying to open a device application on a Windows Mobile device, a Cannot instantiate Ctl_image.FormUntitled1(<i>application name</i>) error is received.	<i>Troubleshooting Windows Mobile Device Applications</i> on page 67
When synchronizing in the child screen of a device application, if the parent data is missing, then old data still displays in the child screen.	<i>Synchronizing Device Applications that Reference Related Mobile Business Objects</i> on page 67
When generating a device application that uses localization in the Device Application Designer, a "Generation Failed: The locale that contains the basic locale of locale must be included " error is received.	<i>Generation Failed Error When Generating a Device Application</i> on page 67
After generating the device application code in the Device Application Designer, launching the BlackBerry Simulator, and synchronizing the device application, a "Missing Sync Parameter" error is received.	<i>Missing Sync Parameter Error Message on the BlackBerry Simulator</i> on page 68

New Features

These new features are included in Sybase® Unwired Platform version 1.2.1.

Creating Multilevel Insert Operations for Web Service Mobile Business Objects

Create a multilevel insert operation for two Web service mobile business objects (MBOs).

In this example, you have two MBOs, Order and OrderItem, that both have defined insert operations: the OrderItem.insert operation requires the Order.id, but Order.id is assigned by the enterprise information system (EIS) and not available until the order is created in the EIS. You can create a multilevel insert operation to address this problem. When creating the multilevel insert operation:

- Ensure that Order.insert operation returns a resultSet that has the newly created Order.Id as one of the columns.
- Chain the two insert operations by creating the appropriate relationship.
- Ensure the association from Order to OrderItem is from Order.id.
- Ensure consistent naming: the **Find By** attribute of Order (ID) must match the ID parameter of OrderItem.insert.

1. Create a Web service connection profile to the data source from which you created the MBOs.
2. Create attributes of the parent MBO (Order). For example, you can drag and drop the Web service data source onto the Mobile Application Diagram, and use the Quick Create wizard to define the MBO.

Define the MBO operation (insert).

Note: Web service multilevel inserts support SOAP bindings only.

3. Click **Finish**.
4. Set or verify the **Fill from attribute** setting:
 - a) In the Mobile Application Diagram, double-click the operation that serves as the insert operation for the parent MBO.
 - b) From the left side of the Properties view, select the **Parameters** tab.
 - c) Verify that each parameter name has a corresponding **Fill from Attribute** value defined.

All parameters of the create operation in the parent MBO and the child MBO must be set to the related **Fill from attribute** value. By default, the related value is set automatically, but in some cases the value cannot be found, so double check the values.

- d) From the left side of the Properties view, select the **Attributes** tab located on the left, then the **Attributes Mapping** tab located on the top. Locate and select the **Find by** check box for the attribute that serves as the primary-key equivalent for the parent MBO (for example, Id).
5. Create the child MBO (OrderItem) the same way you created the parent – drag and drop the data source onto the Mobile Application Diagram, and follow the Quick Create wizard instructions to create the attributes and operations.

6. From the Properties view, verify that each operation's (insert) parameter name has a corresponding **Fill from attribute** setting.
7. In the Mobile Application Diagram, click **Relationship**, and use the wizard to define a relationship between the MBOs. For example, link the Source object Order "Id" attribute to the Target object OrderItem "Id".
Verify that the child MBO is not syncable, unless you are sure that the child MBO will be synchronized either independently, or through the parent MBO. When the device application designed from these MBOs runs, the child MBOs appear on the Synchronize screen. If the device-application user attempts to synchronize any of the child MBOs, a `Missing-Sync-Param` exception occurs.
8. Verify that **Filter by** is selected for the child MBO's attribute/parameter used in the relationship.

Understanding Multilevel Insert Operations

In a multilevel insert, multiple mobile business objects are synchronized in a single operation. The mobile business objects must have a defined relationship, and the insert parameters must support the relationship.

Some business processes require multiple related enterprise information system (EIS) operations; for example, creating a sales order with line items. The parent/child relationship is often represented by primary key(PK) / foreign key(FK) attributes in the parent and child mobile business objects (MBOs). When you construct these types of MBOs in an offline client application, the primary-key and foreign-key values are transitory. When EIS operations are called to create real data, the EIS systems generate the actual key values, and the primary key of the parent is copied to the related child MBO creation operations. These types of operations are known as "chained insert" or "multilevel insert."

- For JDBC MBOs using Sybase databases, dragging and dropping a table that contains autoincrement columns (one mechanism for generating primary keys) automatically creates the appropriate operations for obtaining the parent's generated keys and applying them to the children.
- For other EIS types (non-Sybase databases, and applications where key generation does not use the autoincrement technique), you must define the insert operations in such a way that allows the child to obtain the generated keys.

Typically, in a chained-insert operation, you:

1. Create the parent MBO, and indicate the attributes that constitute that MBO's primary key.
2. Create the child MBO and draw a relationship from the parent MBO's primary-key attributes to the child's foreign-key attributes.

Unselect the child MBO's Syncable property, unless you are sure that the child MBO will be synchronized either independently or through the parent MBO. Otherwise, when the device application designed from these MBOs runs, the child MBOs display on the

"Synchronize" screen. If the device application user invokes "Synchronize" on any of the child MBOs, a `Missing-Sync-Param` exception occurs.

3. Define the insert operations for the parent and child MBOs.

The insert operation for the parent MBO must return a single row that contains the primary-key values. The column labels must match the attribute names of the parent MBO. With this information, and the relationship-mapping data, Unwired WorkSpace modifies the input parameters for the insert operation of the child MBOs by replacing the foreign-key attributes with the ones returned from the parent MBO's insert operation. For example:

```
CREATE TABLE parent(pk int autoincrement primary key, p1
varchar(30), ...)
CREATE TABLE child(fk int references parent.pk, ...)
```

The parent insert MBO is defined as:

```
INSERT INTO parent(p1, ...) VALUES(?, ...); SELECT * FROM parent
WHERE pk = @@IDENTITY;
```

This batch query inserts the new parent row, and returns a single row containing the newly generated primary-key value.

You must understand the key-generation mechanism used by the EIS application from which you are developing, and be able to determine how to retrieve the newly generated keys during the insert operation (frequently, this logic is wrapped in a stored procedure).

This same technique applies to Web service, SAP, and other EIS systems, though the insert-operation definitions differ.

Note:

- The from attribute of the insert operation parameter is used to infer the foreign-key information of the insert operation parameter. So the name of the attribute (which is the target of the association from the primary key of the parent) and parameter of the insert operation need not be the same.
 - The insert query returns the complete newly generated row, not just the identity column. The single row that is returned must contain all of the columns referenced in the relationship between the parent MBO and the child MBO, and the labels of the columns must match the from attribute names of the parent MBO.
Not all columns in the inserted row are required. For example, not all columns are selected or required for a drag-and-drop database operation.
 - A multilevel insert records all logs under the parent MBO. All pending actions are also listed under the parent MBO.
-

Errors may occur if:

- The client sends the parent ID, which does not correspond to the server's interpretation of the parameters of the insert operation.
- The customer's primary key consists of more than one attribute.

If the child has multiple foreign-key attributes pointing to the parent, the relationship should list all relevant parent-to-child attributes. As long as the row returned from the

parent insert contains all those columns, the child insert should work; all the foreign-key fields are populated from the parent insert result set.

- The insert operation of the parent fails at the back end.
- There is no association relationship between customer and order in which the source attribute/parameter in customer is a primary key and the target parameter in order is a foreign key to customer.
- The result set generated by the parent's insert operation does not have the required single row with the newly created primary key of that operation.

Note: Unwired Server does not report the specific reason of a multilevel insert failure. If you receive errors, or if the insert fails, check each of these items to try and identify the problem.

Cache Update Policy

Fine-tune device application and Unwired Server performance by defining a cache update policy for mobile business object operations.

Setting a cache update policy for mobile business object (MBO) operations gives you more control of both Unwired Server interactions with the enterprise information system (EIS) to which the MBO is bound, and consolidated database updates. Fine-tuning these interactions and updates improves both Unwired Server and device application performance.

Note: Consolidated database, CDB, and cache all refer to the same thing, and the terms are used interchangeably.

- MBO operations perform specific functions based on their definition:
 - Primary read operation – the EIS operation used to define and initially populate the CDB (from the EIS) for the MBO.
 - Create, update, delete (CRUD operations) – modify EIS data depending on the definition of the operation. Unwired Server maintains a cache (CDB) of back-end EIS data to provide differential synchronization and to minimize EIS interaction. When an operation is submitted from a device application to the EIS, the cache must be refreshed.

While these types of bulk-fetch and CDB caching are effective in reducing the number of interactions required with the back-end EIS, and work well in some other cases (where MBO data is occasionally updated in the back-end), performance suffers if changes are initiated from Unwired Server (by way of MBO operations), or if changes are frequent.

The cache update policy introduces alternative methods of updating the cache at finer granularity, which improves performance.

- Alternate read operations – can be invoked either from:
 - A chained read cache policy to augment CRUD operations by chaining an alternate read operation to a CRUD operation.

- A data change notification, which provides a mechanism to invoke MBO operations, including alternate read operations. This mechanism is independent of a cache update policy.
- Cache update policy – determines how the CDB is updated after an operation. You can set the cache update policy for operations, with these exceptions:
 - Operations defined as "Other" do not support alternate read or a cache update policy.
 - When invoked, alternate read operations always use the apply operation results cache update policy.

Versions of Sybase Unwired Platform earlier than 1.2.1 supported only the invalidate cache policy—any CUD or other operation issued from a device application invalidated the cache and required a primary read operation to refresh the cache.

In Unwired Platform version 1.2.1, these are the five cache update policies you can associate with MBO CUD operations:

- Invalidate cache
- No invalidate cache
- Apply operation result
- Apply operation parameters
- Chained read

When an MBO CUD operation is called, its cache update policy determines how operation results are applied to the consolidated database. Generally, there are two ways of calling an MBO operation:

1. Device client calls the operation.
2. A data change notification (DCN) request contains the operation.

Note: Other methods used to update the CDBs that are external to MBO operations, and not associated with cache update policies include:

1. EIS-initiated DCN – an HTTP request to Unwired Server, in which the DCN request contains the payload (information about the changed data).

Note: EIS-initiated DCN also supports HTTP POST requests which provides a higher level of security.

2. Scheduled data refresh – defined in Sybase Control Center; polls the EIS for changes at specified intervals.
-

Setting a Cache Update Policy

A Cache Update Policy establishes criteria for updating the Unwired Server cache (also called the consolidate database, or CDB) for a given mobile business object (MBO) operation at a finer granularity than the primary read operation.

Controlling how create, update, and delete (CUD) operation results are applied from the enterprise information system (EIS) to the CDB maximizes efficiency and performance for both Unwired Server, and device applications.

1. Access the **Cache Update Policy** tab from an existing operation by either double-clicking the operation or right-clicking and selecting **Edit**.
2. From the Properties view for the operation, or from the Cache Policy tab in the Operation edit dialog, select the cache update policy:

Cache Update Policy	Description
Apply operation result policy	Updates the CDB based on the returned result set of the called MBO operation.
Apply operation parameters policy	Updates the CDB based on the operation's parameters.
Invalidate cache policy	Invalidates the CDB after the client calls the MBO operation, and requires a primary read operation to fetch all data again from the EIS. This is the default cache update policy.
No invalidate cache policy	The CDB remains unchanged after the client calls the MBO operation.
Chained read policy	<p>Chain an alternate read operation to the MBO operation. The CDB is updated with the results returned by the alternate read operation. Choose an existing alternate read operation or create a new one by selecting Create.</p> <p>The alternate read operation is no different from an operation with an apply operation result cache update policy. Specifying an alternate read type operation enables chaining it to the MBO operations to get the desired result. The difference is that they are used only in conjunction with an MBO CUD operation.</p>

Apply Operation Result Policy

Use the apply operation result policy to update the consolidated database (CDB) based on the returned result set of the called MBO operation.

The operation returns a record set that is processed and applied to the CDB.

This example uses the customer table from the sampled database, which is accessible from the "My Sample Database" connection profile.

1. To create an MBO, drag and drop the sampled customer table onto the Mobile Application Diagram. In the Quick Create wizard, accept the default operations.
2. In the Properties view, double-click the create operation to open the operation.
3. Select the **Definition** tab, and click **Edit** to update the SQL statement to:

```
INSERT INTO sampled.dba.customer
(id,
fname,
lname,
address,
city,
```

```

state,
zip,
phone,
company_name)
VALUES
(@OP["id"= " "],
'@OP["fname"= " "]',
'@OP["lname"= " "]',
'@OP["address"= " "]',
'@OP["city"= " "]',
'@OP["state"= " "]',
'@OP["zip"= " "]',
'@OP["phone"= " "]',
'@OP["company_name"= " "]'
)
SELECT * FROM customer where id = @OP["id"= " "]

```

4. Select the **Cache Update Policy** tab, and select **Apply operation result**.
5. CDB and client data updates are based on the operation definition and the apply operation result policy; other enterprise information system (EIS) changes are ignored.

Apply Operation Parameters Policy

Use the apply operation parameters policy to apply the values of an operation's parameters directly to the cache (CDB).

This example uses the department table from the sampledb database, which is accessible from the "My Sample Database" connection profile.

1. To create a mobile business object (MBO), drag and drop the department table from the sampledb onto the Mobile Application Diagram.
2. Set the update operation's cache update policy to **Apply Operation Parameter**.
3. Deploy the MBO.
4. From Sybase Control Center (SCC) Administration Console, set the MBO's **Cache Interval** to one hour (or any value long enough to complete this test).
5. Sync the MBO from a test client or device application. The client, CDB, and EIS all have the same data:

dept_id	dept_name	dept_head_id
100	R & D	501
200	Sales	902
300	Finance	1293
400	Marketing	1576
500	Shipping	703

6. Call the MBO's update operation from the test client or device application to update this record:

```
dept_id=400, dept_name="QA", dept_head_id=1576
```

The EIS is modified:

dept_id	dept_name	dept_head_id
100	R & D	501

200	Sales	902
300	Finance	1293
400	QA	1576
500	Shipping	703

7. Sync the MBO. Based on the apply operation parameters policy associated with this operation:

1. The dept_id=400 record is updated using the update operation's parameter values (an inferred read), and becomes:

```
dept_id=400, dept_name=QA, dept_head_id=1576
```

2. The CDB and client are updated, resulting in a client and CDB that contains:

dept_id	dept_name	dept_head_id
100	R & D	501
200	Sales	902
300	Finance	1293
400	QA	1576
500	Shipping	703

Invalidate Cache Policy

Use the invalidate cache policy only when other policies cannot be implemented or performance is not an important consideration.

An operation that uses the invalidate cache policy:

1. Performs the create, update, or delete (CUD) operation. For example, insert a new record in the enterprise information system (EIS).
2. Invalidates the cache (CDB) for that mobile business object (MBO) instance.
3. Requires the MBO instance in the CDB to be refreshed from the EIS (this is known as playback). Internally, the MBO's primary read operation executes to retrieve data from the EIS and repopulates the cache.

Invalidate cache is the default cache update policy for any CUD operation for which there is no cache update policy set.

No Invalidate Cache Policy

Use the no invalidate cache policy when changes to the enterprise information system (EIS) do not need to be immediately passed to the consolidated database (CDB).

An operation that uses the no invalidate cache policy updates the EIS without invalidating the cache (CDB) for that instance of the MBO. Because cache is **not** invalidated, it may be different from the EIS. The cache is updated later, for example, based on the **Cache Interval** set on Unwired Server. The no invalidate cache policy eliminates nonessential CDB refreshes, which improves performance.

Note: Consolidated database, CDB, and cache all refer to the same thing, and the terms are used interchangeably.

This example uses the department table from the sampledb database, which is accessible from the "My Sample Database" connection profile.

1. To create a mobile business object (MBO), drag and drop the sampledb's "department" table onto the Mobile Application Diagram.
2. Set the update operation's cache update policy to **No Invalidate Cache**.
3. Deploy the MBO.
4. From Sybase Control Center (SCC) Administration Console, set the MBO's **Cache Interval** to one hour (or any value long enough to complete this test).
5. Sync the MBO from a test client or device application. The client, CDB, and EIS all have the same data:

dept_id	dept_name	dept_head_id
100	R & D	501
200	Sales	902
300	Finance	1293
400	Marketing	1576
500	Shipping	703

6. Call the MBO's update operation from the test client or device application to update this record:

```
dept_id=100, dept_name="SUPQA", dept_head_id=501
```

Sync the MBO. The EIS is modified:

dept_id	dept_name	dept_head_id
100	SUPQA	501
200	Sales	902
300	Finance	1293
400	Marketing	1576
500	Shipping	703

Based on the no invalidate policy associated with this operation, the CDB and client remain unchanged because the cache (CDB) remains valid:

dept_id	dept_name	dept_head_id
100	R & D	501
200	Sales	902
300	Finance	1293
400	Marketing	1576
500	Shipping	703

Chained Read Policy

Use the chained read policy to chain an alternate read operation to a create, update, or delete (CUD) operation. This updates the consolidated database (CDB) at a finer granularity than could be achieved with just the CUD operation.

The parameters of the alternate read operation should be a subset of the parameters of operation to which it is chained. The parameter values of the chaining operation are passed to the alternate read operation, and the results returned from the alternate read operation are applied to the CDB.

This example uses the department table from the sampledb database, which is accessible from the "My Sample Database" connection profile.

1. To create a mobile business object (MBO), drag and drop the sampledb table department onto the Mobile Application Diagram and accept the default operations, when prompted in the Quick Create wizard.
2. Double-click the create operation to open the operation in the Properties view.
3. Select the **Cache Update Policy** tab, and select **Chained read**.
4. Select **create** to launch the New Operation wizard to define and add the alternate read operation:
 - Name – AR
 - Operation type – ALTERNATE READ
 - Specify data source – My Sample Database
 Select **Next** and enter the SQL definition:


```
select * from department where dept_id < 400
```
5. Click **Finish**.
6. Deploy the MBO to Unwired Server.
7. The result is an MBO that when accessed from a device application or test client (assuming that the MBO's **Cache interval** is long enough to complete the test without requiring a cache refresh) behaves as follows:
 - a. The create operation inserts records into the EIS.
 - b. The alternate read operation updates the CDB only for records where dept_id < 400, and is called after the create operation.

From an end-to-end perspective:

1. The initial synchronization of this MBO from a test client or a device application results in the client, CDB, and EIS with the same data:

```
dept_id dept_name dept_head_id
-----
100      R & D       501
200      Sales       902
300      Finance     1293
400      Marketing   1576
500      Shipping    703
```

2. This record is inserted into the EIS (using some method other than the MBO operation):

```
dept_id=1000, dept_name="QA", dept_head_id=501
```

The EIS now contains:

```
dept_id dept_name dept_head_id
-----
100      R & D       501
200      Sales       902
300      Finance     1293
400      Marketing   1576
500      Shipping    703
1000     QA          501
```

3. The client and CDB are not updated with the new record.
4. The client invokes the create operation to insert the record:

```
dept_id=350, dept_name="QA", dept_head_id=501
```

5. The client invokes the alternate read operation and sync's the MBO. While the EIS contains dept_id=1000 and dept_id=350:

```
dept_id dept_name dept_head_id
-----
100      R & D      501
200      Sales      902
300      Finance    1293
400      Marketing  1576
500      Shipping   703
1000     QA          501
350      QA          501
```

Based on the alternate read operation's definition, the only new record retrieved from the EIS and updated in the CDB and client is dept_id=350 record:

```
dept_id dept_name dept_head_id
-----
100      R & D      501
200      Sales      902
300      Finance    1293
400      Marketing  1576
500      Shipping   703
350      QA          501
```

Defining Alternate Read Operations

Define alternate read operations that augment mobile business object (MBO) operations when used with a chained read policy.

A primary read operation is the enterprise information system (EIS) operation for an MBO that defines how the MBO is populated. In Sybase Unwired Platform 1.2.1, a new operation of type "read" has been introduced. A read operation, also called an alternate read operation, always uses the apply operation result cache policy when it returns a record set that is applied to the consolidated database (CDB, or cache). You can define any number of alternate read operations for an MBO. In general alternate read operations return data corresponding to a finer granularity than a primary read of the MBO, enabling the cache to be updated at a finer granularity. Alternate read operations can be chained to any create, update, or delete (CUD) operation of an MBO. The Device API does not directly support calling the alternate read operations, but you can invoke them independently using data change notification (DCN) requests.

1. Define an alternate read operation using one of these methods :
 - Select the Operation icon from the Mobile Application Diagram palette and click the Operations portion of the MBO.
 - Select Add from the Operations tab of the Properties view.
 - You can also create alternate read operations when you assign a chained read cache policy to an MBO operation.
2. From the New Operation wizard, name the operation, and specify the Alternate Read as the **Operation type**.

3. Specify the data source type and connection profile from which you are creating the operation, and click **Next**.
4. Complete the operation definition according to the data source type to which you are binding the operation.

For example:

- Database datasource – select any of the operations.
 - Web service datasource – from the XSLT Definition screen, select **Configure XSLT** to access the XSLT.
5. Modify the operation to meet the intended need. For example, you may have an MBO operation that inserts a record into the database, then the alternate read operation retrieves the record only if it affects a particular user.

Alternate Read and Cache Update Policy Validation Rules

Validation rules are enforced when you define an alternate read operation, set a cache update policy, and, in some cases, when you deploy the mobile business object (MBO) to Unwired Server.

Table 1. Alternate read and cache update policy validation error messages

Severity and message	Description	Occurs when:
Error: No "Find By" attributes set for the mobile business object, cannot create alternate read type operation.	While a primary key for the data source is not necessary (Web service data sources do not have primary keys), the MBO must have Find by set for at least one attribute.	Creating an alternate read operation
Error: The columns of the alternate read operation "{0}" must contain the "Find By" attributes of the associated mobile business object.	The result set columns in the alternate read operation do not contain all the necessary Find by attributes.	Creating an alternate read operation
Error: Cache update policy of operation "{0}" is set to "{1}", but there are no attributes with "Find By" set for the mobile business object.	The MBO must have at least one Find by attribute.	When setting an operation's cache update policy to apply operation result, apply operation parameter, or chained read

Severity and message	Description	Occurs when:
Error: Operation "{0}": when the cache update policy is "Apply Operation Parameters", the operation parameters' "Fill From Attribute" must have "Find By" set.	The Fill from attribute property of the operation's parameter must contain all of the MBO's key attributes (Find by attributes). This can be done in Parameter tab of the Operation's Properties view.	Setting the cache update policy to apply operation parameter
Error: Must specify an alternate read operation for operation "{0}" when the cache update policy is "{1}".	No alternate read operation is specified in a CUD operation whose cache update policy is "chained read."	Defining a create, update, or delete (CUD) operation's cache update policy to chained read, or when a chained alternate read operation is deleted
Error: Parameters of the chained operation "{0}" should be a subset of the parameters of the chaining operation "{1}".	The parameters of the chained operation must be a subset of the parameters of the chaining operation.	Selecting an alternate read operation while defining a chained read policy
Warning: The columns of the read operation "{0}" must cover 'filter by' parameters of the associated mobile business object.	The resultset of the alternate read operation does not contain all of the MBO's "Filter by" attributes or parameters.	Defining an alternate read operation
Warning: Alternate read operation "{0}" must cover all mobile business object attributes with "Filter By" set.		

Alternate Read Requirements

Mobile business objects must meet certain requirements before you can add alternate read operations to them.

Before you can add an alternate read operation to a mobile business object (MBO), the MBO must meet these requirements:

- The MBO must be bound to a data source that has one or more Find by attributes set.
- All columns in the record set returned by the alternate read operation should contain all key attributes.

- The result set of the alternate read operation must contain all MBO playback and sync parameters (Filter by parameters and attributes).

Alternate read operations are filtered from MBO code and not passed to the Device Application Designer.

Covering the Playback and Synchronization Parameters of a Primary Read Operation

The result set of an alternate read operation must contain the playback and sync parameters of the mobile business object (MBO).

For an alternate read to be valid, its result set must include any MBO playback and sync parameters.

For example, create three MBOs by dragging and dropping the `sampledb.dba.customer` data source onto the Mobile Application Diagram. Modify the MBO definition with the following SQL statement, and set Filter by for both the "state" parameter and the "company_name" attribute.

```
SELECT id, fname, lname, address, city, state, zip, phone,
company_name FROM
sampledb.dba.customer WHERE state=@OP["state"="CA"]
```

This simple example illustrates a validation error. Create an alternate read operation for each MBO using the SQL statements:

- Customer1 MBO

```
select * from customer
```

This is valid, since the alternate read result set contains the "state" and "company_name" playback and sync parameters.

- Customer2 MBO

```
select id, fname, lname, phone, company_name from customer where
company_name=@OP["company_name"="AAA"]
```

A warning displays since "id, fname, lname, phone, company_name" does not contain the "state" playback and sync parameter.

Note: Data in the @OP clause is not used for validation.

- Customer3 MBO

```
select id, lname, fname, phone, company_name from customer where
state=@OP["state"="CA"]
```

This query also generates a warning. Although the read operation's parameter contains "state," the result set does not contain the "state" playback and sync parameter. It contains another sync parameter "company_name."

Adding an SAP Result Checker

Create or add an SAP result checker when you edit Attribute or Operation properties for a mobile business object derived from an SAP data source under the Definitions tab. You can

also configure SAP result checkers when you create an object. You can choose a predefined or a custom SAP result checker, if you have created one.

1. In the New Attributes or New Operation wizard, in the Result checker section, select from these options:

Option	Description
Default	If there is a RETURN parameter found in the currently selected BAPI operation, this option is automatically selected. This option uses the com.sybase.sap.DefaultSAPResultCheck result checker.
None	If there is not a RETURN field found in the currently selected BAPI operation, this option is automatically selected. This option uses the com.sybase.sap.NoOpSAPResultCheck result checker.
Custom	Select this option to specify a custom SAP result checker.

2. (Optional) If you have not yet created the result checker classes, then in the Result checker area of the New Attributes or New Operation dialog, select **Custom** and click **Create** to run the New Java Class wizard.

3. If prompted, add a Java nature.

- a) Click **Yes** to add a Java nature. In Eclipse, a Java nature adds Java-specific behavior to projects. A Java nature is recommended because you are adding a Java class to it. By default Unwired Platform projects do not include all the required behaviors for Java development.

In the New Java Class wizard, enter:

Option	Description
Source Folder	By default, this is the Filters folder from your project. Click Browse to locate the source folder for the Java class.
Package	Click Browse to locate the package for the new Java class. Note: Sybase recommends that you specify the package and do not leave this field blank as the JDK 1.4 Java class in the default package cannot be resolved in other packages.

Option	Description
Enclosing type	Select to choose a type in which to enclose the new class. You can select either this option or the Package option, above. Either enter a valid name or click Browse .
Name	Enter a name for the SAP result checker class.
Modifiers	Select the Java class modifiers.
Superclass	By default, this is filled in with java.lang.Object. <ol style="list-style-type: none"> Click Browse. In the Superclass Selection dialog, enter: <ul style="list-style-type: none"> Choose a Type Matching Items Click OK.
Interfaces	By default, this is populated with the com.sybase.sap.SAPResultChecker interface. Click Add to select interfaces implemented by the new class.
Which Method Stubs Would You Like to Create	<ul style="list-style-type: none"> Public Static Void Main Constructors From Superclass Inherited Abstract Methods
Do You Want to Add Comments	Select Generate Comments to add comments. <hr/> Note: If you set the Java project compiler compliance level to later than 1.4, you receive comments about missing classes. Set the Java project compiler compliance level to 1.4 or earlier. Java projects that are configured to compile with a compiler compliance level later than 1.4, have a known issue, where Java files referring to the RIM API have missing class errors.

Note: The wizard generates a skeleton Java source file only.

- Click **No** if you do not want to add the Java nature to the selected Mobile Application Project.
- Click **Finish** in the wizard to compile the java skeleton source file and add the skeleton java checker class to the MBO.

The SAP result checker you created appears next to the Custom option.

- In the Result checker section, next to the Custom option, click **Search** to find an existing SAP result checker class.

- a) In the Select SAP Result Checker Class dialog, select the SAP result checker class and click **OK**.

The SAP result checker class appears next to the Custom option.

Note: A valid SAP result checker is a Java class that implements `com.sybase.sup.sap.SAPResultChecker`. Only classes implementing the correct interface appear in the search dialog.

5. Implement the new class by writing the implementation on top of the skeleton, as documented in the topic, *Writing a Custom SAP Result Checker*.
6. Test and preview the result of your result checker:
 - a) To reuse input values you have already saved for previous previews, select **Existing Configuration**. Otherwise, load defaults, or create a new set of input values expressly for this preview instance.
 - b) Click **Preview**.

If the data runs successfully, `Execution Succeeded` appears at the top of the Preview dialog and data appears in the **Preview Result** window.

Writing a Custom SAP Result Checker

An SAP result checker is a custom Java class that implements error checking for SAP mobile business objects.

Since not all SAP BAPIs or RFCs use the "standard" error reporting technique, you can implement your own custom SAP result checker. This allows you to check any field for errors, or implement logic that determines what constitutes an error, and the severity of the error.

1. Provide a Java class that implements the `SAPResultChecker` interface:

```
package com.sybase.sup.sap;
public interface SAPResultChecker
{
    /**
     *
     * @param f - JCO function that has already been executed.
     * Use the JCO API to retrieve returned values and determine if
the RFC has executed
     * successfully.
     * @return a single Map.Entry. The boolean "key" value should
be set to true if the
     * RFC is deemed to have succeeded. Normal result processing
will ensue.<P>
     * If the String value is not empty/null, that value will be
treated as a warning message,
     * which will be logged on the server,
     * and returned as a warning in transaction logs to the
client.<P>
     * Set the key value to false if it is deemed the RFC has
failed. The String value will
     * be thrown in the body of an exception. The error will be
logged on the server, and the
```

```

    * client will receive a transaction log indicating failure,
    including the string value.
    */
    Map.Entry<Boolean, String> checkReturn(JCO.Function f);
}

```

There are two `SAPResultChecker` implementations located in `%SUP_HOME%\Servers\UnwiredServer\uep\tomcat\webapps\onepage\samples\sap` directory, which you can modify and reuse as custom result checkers:

- `DefaultSAPResultCheck` – the default SAP result checker.
 - `NoOpSAPResultCheck` – this result checker always returns a successful result.
2. Save any classes you create to an accessible Unwired Workspace location. This allows you to select the class when you configure the SAP result checker for your mobile business object.

Deploying SAP Result Checker Classes to Unwired Server

Before deploying SAP mobile business objects that use result checker classes, copy the compiled classes to `<Unwired-Server-install>\lib\filters\<packageName>` on the primary Unwired Server machine.

You can also place result checker classes into a directory structure; the root of which should start at `packageName`. During cluster synchronization, the SAP result checker classes are automatically distributed to other Unwired Servers in the cluster.

1. (Optional) To maintain and deploy a single file, create a Java archive of all your classes.
2. Choose the Unwired Server target location for the result checker classes, dependencies, and third-party JARs:
 - To avoid restarting Unwired Server, copy either the JAR file or the individual class files to:


```

          <Unwired-Server-install>\lib\filters
          \<DeploymentPackageName>.
          
```

 For example, if you have a `ResultChecker_1.0.0` package and a `com.acme.filters.ResultChecker` result checker class, create


```

          <Unwired-Server-install>\lib\filters
          \ResultChecker_1.0.0\com\acme\filters
          \ResultChecker.class. Or, create acmeFilters.jar from the
          com.acme.filters.ResultChecker classes, and copy it to <Unwired-
          Server-install>\lib\filters
          \ResultChecker_1.0.0\acmeFilters.jar.
          
```
 - To restart Unwired Server after deployment, copy the classes to:


```

          <Unwired-Server-install>\lib\filters\
          
```
3. Deploy the SAP mobile business object to Unwired Server.

Editing the SAP Result Checker

Change the SAP result checker settings using the Change Definition dialog.

1. Right-click inside the Mobile Application Diagram and select **Show Properties View**, or select **Window > Show View > Properties**.
2. In the Properties view, click the **Attributes** or **Operations** tab, then the **Definition** tab.
3. Click **Edit**.
4. Make your changes in the Change Definition dialog, and click **OK**.

Refactoring an SAP Result Checker

When an SAP result checker is deleted, renamed, or moved, update its references automatically.

Deleting References to an SAP Result Checker

Delete all references to an SAP result checker from the workspace.

1. In WorkSpace Navigator, select the mobile application project that contains the SAP result checker, and expand the **Filters** folder.
2. Right-click the SAP result checker and select **Refactor > Delete**.
3. In the Confirm Delete dialog, verify the selected references, and click **OK**.

Renaming an SAP Result Checker

Rename an SAP result checker, and update its references in the workspace.

1. In WorkSpace Navigator, select the mobile application project that contains the SAP result checker you want to rename, and expand the **Filters** folder.
2. Right-click the SAP result checker, and select **Refactor > Rename**.
3. In the Rename Type dialog, verify the changes, and click **Finish**.

Moving an SAP Result Checker

Move an SAP result checker to another location, and update its references in the workspace.

1. In WorkSpace Navigator, select the mobile application project that contains the SAP result checker you want to move, and expand the **Filters** folder.
2. Right-click the SAP result checker, and select **Refactor > Move**.
3. In the Move dialog, confirm the changes, and click **OK**.

Searching for References to an SAP Result Checker

Search for mobile business objects that reference SAP result checker classes.

SAP mobile business objects can share SAP result checker classes, within a single project or across multiple projects.

1. In WorkSpace Navigator, in the mobile application project, expand the **Filters** folder.

2. Right-click the SAP result checker, and from the context menu, select **References > Mobile Business Object**.

The search results appear in the **Search** pane.

Importing Visual Studio Projects into Eclipse

Use the native Eclipse Import feature to import exported Visual Studio Unwired WorkSpace projects into Unwired WorkSpace (Eclipse).

1. Select **File > Import**.
2. Expand the General folder, select **Existing Projects into Workspace**, and click **Next**.
3. Select the root directory—or archive file—that contains the projects you are importing.
4. Projects display in the **Projects** section. Select the projects you want to import, select **Copy projects into workspace**, and click **Finish**.

The imported projects appear in WorkSpace Navigator.

Updating Imported Projects

After importing a Visual Studio project into Eclipse, you must manually perform several tasks. This is due to differences between Visual Studio and Eclipse models (for example, different levels of feature support for each platform), after importing a Visual Studio project into Eclipse you must manually perform some steps to correct warnings and errors.

Issue	Solution
Web service method information is not imported.	Manually define the Web service methods after importing the project.
SAP BAPI definitions are not imported.	Manually set the BAPI definitions after importing the project.

Configuring the SAP AutoCommit Feature

Configure an SAP operation, so `commit` is always called after the operation succeeds.

For an SAP operation, if the AutoCommit feature is enabled (`requireCommit` property is set to true), `commit` is always called following a successful operation; if the operation fails, changes are rolled back. By default, the AutoCommit feature is enabled; if disabled, you must explicitly call `commit` after the operation succeeds.

1. In the Mobile Business Object Properties dialog, select **Attributes or Operation**, then click the **Definition** tab.
2. Click **Edit**.
3. To enable the AutoCommit feature, check **Commit SAP Operation**; to disable, uncheck **Commit SAP Operation**.
4. Click **OK**.

Modifying SAP Connection Properties

Use SAP connection properties as mobile business object (MBO) parameters when making connections from an SAP MBO to an enterprise information system (EIS).

Modify SAP Java connector (JCo) connection properties when creating or editing an SAP mobile business object. You can then use these connection properties as parameters, for example, as personalization keys or default values.

Modify connection properties either from the Mobile Business Object Creation wizard or the Properties view.

1. In the Definition Window of the Mobile Business Object Creation wizard, expand **Runtime Data Source Credential and Connection Properties**.
2. The Connection Properties table displays the configurable connection properties. The Property column is read-only, but you can modify SAP JCo connection properties:
 - You can either input a value or select a personalization key from the drop down list for the property. You can also create a new personalization key for any property except the language property, which does not support personalization.
 - The codepage property does not appear in the table; it is calculated from the language and cannot be modified.
 - Set the Unicode property to either true or false.
3. To modify connection properties after the MBO is created:
 - a) From the Properties view, select the **Attributes** tab on the left, then the **Definition** tab.
 - b) Click **Edit**.
 - c) Modify as required and click **OK**.

Once you have created or modified your MBO, you can use these parameters as default values and personalization keys as needed.

At runtime, you can manage SAP connection properties as parameters. For example, if you have an SAP **SalesOrder.CreateFromData1** operation that inserts a sales order for a particular user that occurs in the context of a known SAP user, the user's credentials can be used in the insert operation.

Documentation Updates

These documentation updates apply to Sybase Unwired Platform version 1.2.1.

Playback and Synchronization Parameters

Combine playback (refresh) and synchronization (sync) parameters to control the way that data is cached in the consolidated database (CDB) in Unwired Server, and filtered and returned to a device application.

Mobile business object (MBO) playback and sync parameters affect the data returned from the enterprise information system (EIS) to the Unwired Server cache, called the consolidated database, or CDB, as well as filtering the data before it is returned to the device application.

Parameters	Result
Playback=none sync=none	The entire table is downloaded to the device. For example: <pre>select * from sampledb.dba.customer</pre>
Playback=none sync=region	Only customers of a specific region are downloaded to the device. Typically, region is paired with a personalization key. For example, a sales representative living and working in the western region is interested only in customers from that region, which can also be set as the default value: <pre>select cust_id, cust_name, region from sampledb.dba.customer where re- gion=@OP["region"="western"]</pre>
Playback=region sync=region	Occurs if customer playback requires a region parameter (which is also the sync parameter). This scenario is more likely for Web service and SAP MBOs than for database MBOs.
Playback=username, password sync=region	Playback only for a particular user (with proper authentication) using the username and password parameters, but synchronize based on the region: <pre>select cust_id, cust_name, region from sampledb.dba.customer where re- gion=@OP["region"="western"], for user A.</pre>

Overriding the Default Filter by Setting

You may want to override the default attribute/parameter Filter by setting defined in Unwired Workspace.

By default:

- Filter by is unselected for profile arguments.
- Filter by is selected for non profile arguments.

You may want to change the default Filter by setting after dragging and dropping a data source to create the MBO, then editing the MBO definition by adding any result-affecting parameter (which must have Filter by selected), for example:

```
select * from sampledб.дба.sales_order where sales_rep =
'@OP["sales_rep"]=" "]
```

Since different values of "sales_rep" generate different result sets, the parameter "sales_rep" is result-affecting.

An operation that includes user name and password parameters does not affect the results; in this case leave Filter by unselected.

Note: Sometimes profile parameters affect results, for example, a Web service that returns different results for different users. Unwired Server cannot determine (either at deployment or runtime) whether a parameter is incorrectly marked, you must override the default.

In summary, when developing MBOs, you must select Filter for MBO parameters that affect results and leave it unselected for MBO parameters that do not affect results. Do not change the default Filter by setting for an MBO parameter unless you have a profile parameter for which different values of the parameter return different results.

Table 2. Attribute/parameter usage and Filter by settings

Type	Playback?	Sync?	Description
Attribute	No	Yes	An attribute is a sync parameter and the table must be filtered further by that column for downloading to the device.
Profile parameter	Yes	No	<p>A playback parameter is not a sync parameter if it does not affect the playback results. This is typical of profile parameters such as user name and password. However, some profile parameters, such as an e-mail object, do affect results.</p> <p>When developing MBOs, profile parameters that do not affect playback results have a default setting of unselected for Filter by. If a different login produces different results, unselect Filter by.</p>

Type	Playback?	Sync?	Description
Operation parameter	Yes	Yes	<p>Playback parameters, for example region, almost always affect the results and as such are also sync parameters.</p> <p>Operation-specific parameters, such as a custId parameter in an orders MBO, are invariably result-affecting. The default setting for Filter by for these parameters is selected, and there is rarely a reason to change that setting.</p>

An example of an operation parameter: Say you have a "WorkOrder" MBO that includes an "assigned_to" parameter and a "customer" parameter. Users can search for work orders that have been assigned to themselves, but only to retrieve work orders related to a particular customer. While defining this MBO:

- Select the "assigned_to" parameter as Filter by because it partitions the "WorkOrder" MBO data by user.
- Associate the "login_name" personalization key to this parameter so it does not have to be explicitly passed from the client for each sync.

If you want the device application to have work order data for only one customer at a time, unselect Filter by for "customer". Each time a user changes their customer parameter, Unwired Server reads back work orders for only that customer. Since the work orders are all different from the previous customer, a differential calculation deletes all previous work order rows in the Consolidated Database (CDB) for the current user, and inserts new rows with the new work orders. The differential sync deletes all old rows and insert new ones on the client.

If you unselect Filter by for both the user and customer, the entire CDB table is emptied each time a user syncs. This behavior is almost never desired.

Preparing the Unwired Platform Environment for SAP Connections

The SAP JCO connector is used by all Unwired Platform components. Therefore, Sybase recommends that you make all changes concurrently in a distributed environment for development and production installations of Unwired Platform.

Prerequisites

Before you can access the SAP Web site, you must have an SAP account.

These steps describe the common tasks for setting up an SAP environment. Other details, such as where to copy files, differ by component.

1. Go to the SAP Web site at <http://service.sap.com/connectors> and download the latest SAP JavaConnector, for example, sapjco-ntintel-2.1.8.
2. Unzip the file, which contains:

New Features 1.2.1

- sapjco.jar
 - librfc32.dll
 - sapjcorfc.dll
3. Shut down all Unwired Platform components, including Unwired WorkSpace and all Unwired Servers on your network.
 4. Copy librfc32.dll and sapjcorfc.dll into the following target directories:

Component	Targets
Eclipse Unwired WorkSpace	<ul style="list-style-type: none"> • C:\WINDOWS\system32 • <SUP Installation root>\Unwired-Platform\JDK1.6.0_12\bin
Visual Studio Unwired WorkSpace	<ul style="list-style-type: none"> • \Program Files\Microsoft Visual Studio 9.0\Common7\IDE • <SUP Installation root>\Unwired-Platform\Unwired_WorkSpace\VisualStudio\toolingapi\lib
Unwired Server	<ul style="list-style-type: none"> • <SUP Installation root>\Unwired-Platform\Servers\UnwiredServer\dll

5. Copy sapjco.jar into the following target directories:

Component	Targets
Eclipse Unwired WorkSpace	<ul style="list-style-type: none"> • <SUP Installation root>\Unwired-Platform\Unwired_WorkSpace\Eclipse\sybase_workspace\mobile\eclipse\plugins\com.sybase.uep.com.sap.mw.jco_1.2.0.<version>\lib
Visual Studio Unwired WorkSpace	<ul style="list-style-type: none"> • \UnwiredPlatform\Unwired_WorkSpace\VisualStudio\toolingapi\lib
Unwired Server	<ul style="list-style-type: none"> • <SUP Installation root>\Unwired-Platform\Server\UnwiredServer\lib

6. In a clustered production environment, you must also enable SAP mobile business objects to connect to an SAP R/3 system that uses a router:
 - a) Change to \UnwiredPlatform-1_2\Servers\UnwiredServer\Repository\Instance\com\sybase\sap.
 - b) In a text editor, open <SAPprofile>.properties, where *SAPprofile* is the name of the connection profile that is used to deploy SAP mobile business objects from Unwired WorkSpace to the server.

c) Set the value of the `jco.client.ashost` property to `/H/proxyHost/H/applicationServer`, where:

- `proxyHost` is the name of the proxy-server machine, and
- `applicationServer` is the name of the application server

7. After copying the files, restart all components and all Unwired Servers.

Configuring Your Environment to Use a JDBC Driver

Download the appropriate JDBC driver and configure your environment to connect to Oracle, DB2, and SQL Server databases.

1. Download the driver.

JDBC driver for:	URL
Oracle	http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html
DB2	http://www-306.ibm.com/software/data/db2/express/download.html
SQL Server	http://msdn.microsoft.com/en-us/data/aa937724.aspx

2. For Unwired WorkSpace, put the driver in the correct location.

JDBC driver for:	Action
Oracle	Place the JDBC driver in: <pre><SUP installation root>\Unwired_WorkSpace\Eclipse\sybase_workspace\mobile\ eclipse\plugins\com.sybase.uep.com.oracle.<plugin version number>\lib</pre>
DB2	Unzip the <code>db2JdbcJars.zip</code> file and copy the JAR files to: <pre><SUP installation root>\Unwired_WorkSpace\Eclipse\sybase_workspace\mobile\ eclipse\plugins\com.sybase.uep.com.db2_1.0.0.<timestamp>\lib</pre>
SQL Server	Copy the <code>sqljdbc.jar</code> file to: <pre><SUP installation root>\Unwired_WorkSpace\Eclipse\sybase_workspace\mobile\ eclipse\plugins\com.sybase.uep.com.sqlserver_1.0.0.<timestamp>\lib</pre>

3. Copy the appropriate JAR file to the specified server location.

JAR file for:	Action
Oracle	Copy ojdbc14.jar to the server location: <SUP Installation root>\UnwiredPlatform\Servers\UnwiredServer\lib
DB2	Copy the JAR files to the server location: <SUP Installation root>\UnwiredPlatform\Servers\UnwiredServer\lib
SQL Server	Copy the JAR files to the server location: <SUP Installation root>\UnwiredPlatform\Servers\UnwiredServer\lib

Note: If you do not copy the JAR files to the server location, you will encounter runtime errors due to the missing JDBC driver.

4. Restart Unwired Server.
5. If Unwired Server is running as a Windows service:
 - a) Shut down the server.
 - b) Open a command window and change to `<installation_directory>\Servers\UnwiredServer`, and run:


```
mlservice.bat install
```
 - c) Restart the service.

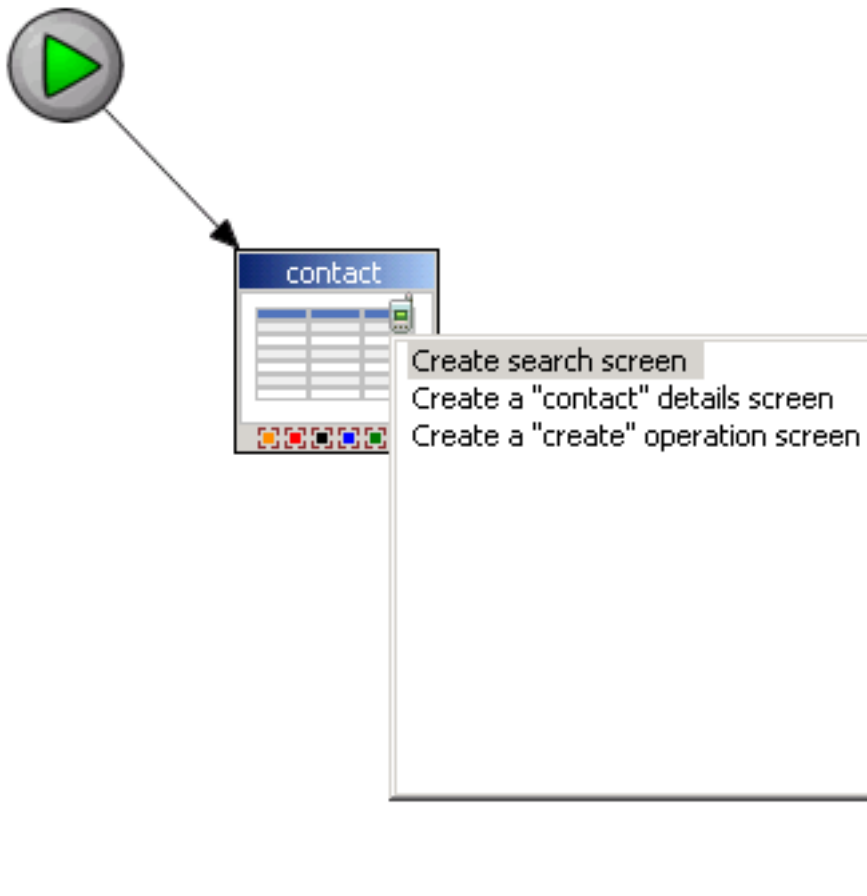
Device Application Designer

These Device Application Designer topics have been updated for Unwired Platform 1.2.1:

Using Drag and Drop to Add Mobile Business Objects to the Flow Design

Drag and drop mobile business objects onto the Flow Design canvas to create screens that are populated with data from the mobile business object.

1. In the Device Application Designer, open the **Flow Design**.
2. In WorkSpace Navigator, locate the mobile business object for which you want to create a screen, select the mobile business object and drag and drop it onto the **Flow Design** page. Screens and connections are created automatically for the mobile business object if the **Automatic Creation** preference in the **Preferences > Device Application Designer** dialog is selected, which, by default, it is.
3. You can manually create screens by clicking the mobile business object icon in the upper-right corner of the initial screen to see the options for creating additional screens. Double-click your selection to create the screen.



Introduction | Flow Design | Screen Design | Source

The screen is added to the Flow Design canvas.

Automatic Screen Creation

Automatically create screens that are populated with data from the data source by dragging and dropping a mobile business object onto the Flow Design.

Set the preferences for automatic screen creation in the Device Application Designer Preferences. These preferences allow you to decide which screens to create when you drop a mobile business object onto the Flow Design canvas.


The Relationship Automatic Screens option means that all sub screens are created after the initial relationship screen is created.

By default, Delete Operation Screen is unselected, which means the Device Application Designer:

- Creates the main screen with a table for the mobile business object.
- Adds a Delete menu item to the main screen.
- Adds an alert action of the question type, which goes back to the main screen to the menu item. Adding this action means the user is prompted to confirm before deleting.
- Adds a save context action to the menu item.
- Adds an operation action that performs the delete operation to the menu item.

If Delete Operation Screen is selected, a Delete screen is created.

If you drop more than one mobile business object onto the Flow Design canvas, only the main screens are created. If a screen has multiple related screens that can be created, use the assist

option  at the top of the screen to create the screens.

Screen Design Palette Options

Design the screen for a device application.

Controls

Option	Description
Header	Create a header on the screen.
Footer	Create a footer on the screen (if there is no header on the screen, creating a footer automatically creates a header as well).
Label	Display text.
Edit Box	Enter a line of text and assign values to strings, dates, decimals, integers, and so on.
Choice	Display a collection of items as a drop-down list.

Option	Description
Hyperlink	Display a label that is underlined like a link and that can be attached to an action for screen transitions.
Checkbox	Assign or display Boolean values.
Spacer	Create a defined space on the screen. This allows for vertical spanning as well as horizontal spanning.
Radio Button	Assign predefined options.
List Item	Similar to buttons, list items are grouped in the List Group container and can have styles and actions assigned to them.
Button	Trigger actions such as screen transitions.
Image	You can place an image into the display as widgets (button, label, and so on). There are two types of images—one type uses the image references to display the image, the other type uses table context variables that are of image types. You can also assign actions to images.
Separator	Visually isolate parts of the screen.
Table	The existing table widget that displays grid data in a table.
List Detail	Show property details for columns.

Containers

Option	Description
Region	A general purpose container that can be used to group any controls.
Radio Group	A region in which to group radio buttons.
List Group	A container for list items.
Tab Folder	A container for tab panels.

Option	Description
Tab Panel	Used to organize information in a tabular format. For example, you can break a long form into sections represented by tab panels. Table and List Detail controls that are related to each other can also be grouped with tab panels.

Menu

Option	Description
Menu Item	Performs the assigned action, for example Refresh or Synchronize.
Menu Separator	Drag and drop menu separators from the Palette onto the screen menu to visually organize the menu using horizontal lines. You can use multiple menu separators, which you can move and delete. Menu separators are not tied to actions.

Actions

Option	Description
Alert	Alerts the user. You can create new alerts using the Alert Action dialog.
Connection	Navigates to a different screen. <hr/> Note: If there are multiple actions assigned to one control or menu, the connection action must be the last action performed. <hr/>
Exit	Adds the exit action to the action list, which exits the client application.
Logout	Logs the user out of the client.
Persist	Works with user variables created in the Variable section. Allows you to save values from the screen controls or literals to set values on variables created in the Variable section when the action is performed.

Option	Description
Refresh	Adds a refresh action to the list of actions, which refreshes the current screen based on updated data. You may want to refresh, for example, after you have synchronized a mobile business object.
Operation	Enables you to create an action based on the operation of a mobile business object.
Synchronize	When the action is executed, the associated mobile business object is synchronized.
Tab Activation	<p>Brings a specified tab to the front. There are three modes for the tab activation action:</p> <ul style="list-style-type: none"> • Specific • Previous • Next <p>The tab activation action can wrap, which means if the current tab is the first tab, clicking the Previous tab goes to the last tab.</p> <p>When the current tab is the last tab, clicking the Next tab goes to the first tab.</p>
Save Context	Adds an action to the menu so that context is maintained when navigating between screens.
BlackBerry PIM	This action is available only if the selected platform for the screen is BlackBerry. Adding the BlackBerry PIM action to a control with the logical type assigned allows you to fully integrate BlackBerry applications such as calendar, address book, tasks, and memo.

Validation


Assign rule validation to the Edit Box control to validate user input.

Option	Description
Rule Validation	<p>Drag and drop onto the Edit Box control to add pattern-matching rule validation. You can assign these pattern-matching datatypes:</p> <ul style="list-style-type: none"> • Numerical • String • Date/Time

Choice Properties

Use the Choice control to display a collection of items as a drop-down list.

Table 3. Choice control properties

Property	Description
Mobile Business Object	Click Search to find the mobile business object with which to associate the selected control.
Display Name Attribute	This option is enabled when this control is associated with a mobile business object. It is populated with the selected mobile business object attributes. This provides content to the control during run time.
Value Attribute	This field is enabled when the Mobile Business Object field is not empty. It is populated with the mobile business object attributes. By default, the first attribute is selected. This attribute can be the same as the Display Name Attribute.
Linked Value	<p>Enter a value for linking this choice to other choice elements on the screen. The value must be from the data available in the "display_name" attribute of the mobile business object.</p> <p>The linked value is cleared and disabled if the mobile business object does not contain the necessary attributes with which to link parameters.</p> <p>The device display shows the link decorator  for any choice fields that are linked.</p>
Items	<p>Click Add to add the items for the choice drop-down list.</p> <p>Click Edit to edit the selected item.</p> <p>Click Remove to remove the selected item.</p>

Property	Description
Initial Value	Select the initial value for the choice drop-down menu. The choices in this drop-down list are set in the Items section.
Use Variable	Use a variable for the Choice text. Click Browse to find available variables.
Read Only	Select if the choices displayed on the screen should be read-only.
Data Type	Select the data type from the drop-down list. Use this option on the device to give menu assistance to inputs, for example, selecting a calendar for date. The device retrieves the information and performs validation on the input, for example, INT must be a number. It is also used to map the control to the operation parameter correctly.
Logical Type	Sets the logical type for the choice. This is used on the device to give menu assistance to input fields, for example, task, calendar, e-mail, or phone.
Style	Select the choice style from the drop-down menu, or click Search to search available styles. Click Edit to edit the selected choice style. Click Clear to clear the choice style.
Span	Choose: <ul style="list-style-type: none"> • Horizontal – enter the number of columns the widget will occupy. The number must be greater than zero, and less than or equal to the number of columns in the screen display. • Vertical – enter the number of rows the widget will occupy.

Radio Group Properties

Use the Properties page to edit the attributes for the Radio Group container control.

Property	Description
Number of Columns	The number of columns in the radio group. You can separate tabs in to 1 – 3 columns. The default is 1.

Property	Description
Assign Percentages	<p>The width of each column in the radio group, in percentages. The defaults are:</p> <ul style="list-style-type: none"> • 1 Column – 100%. • 2 Columns – 35% for the first column, 65% for the second column. • 3 Columns – 33% for the first column, 34% for the second column, and 33% for the third column.
Select Radio	Select the radio button to set initially. The selected radio button appears as selected on the device display.
Use Variable	Select to associate the selected radio button with an existing variable so that all the radio buttons on the display appear disabled. Click Browse to find existing variables.
Data Type	Select the data type from the drop-down list. Use this option on the device to give menu assistance to inputs, for example, selecting a calendar for date. The device retrieves the information and performs validation on the input, for example, INT must be a number. It is also used to map the control to the operation parameter correctly.
Logical Type	Sets the logical type for the choice. This is used on the device to give menu assistance to input fields, for example, task, calendar, e-mail, or phone.
Span	<p>Choose:</p> <ul style="list-style-type: none"> • Horizontal – enter the number of columns the radio group will occupy. The number must be greater than zero, and less than or equal to the number of columns in the screen display. • Vertical – enter the number of rows the radio group will occupy.

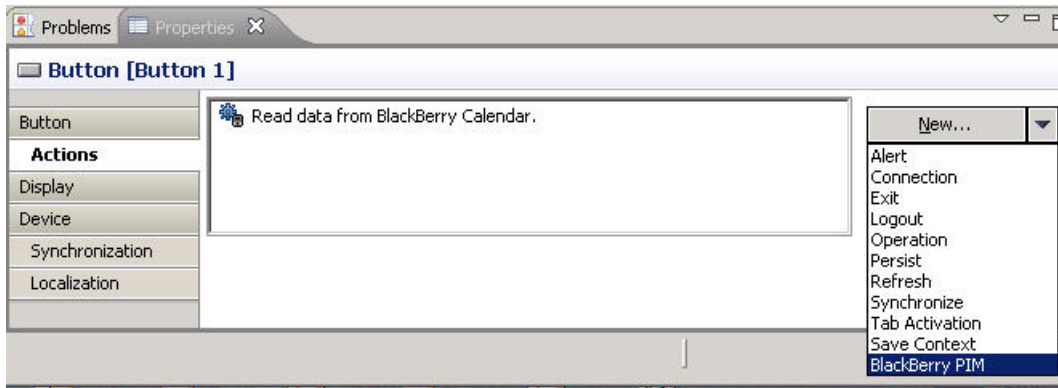
Adding a BlackBerry PIM Action

Add the BlackBerry PIM action to a control to select a BlackBerry PIM application.

You can assign only one BlackBerry PIM action per control.

1. Drag and drop the BlackBerry PIM action from the Palette onto a control or menu item that accepts an action. You can also add the BlackBerry PIM action using the Properties view for the selected control:

- a) Select the control to which you want to add the BlackBerry PIM action.
- b) In the Properties view for the control, select **Actions** from the left pane.
- c) In the Actions dialog, click **New** and select **BlackBerry PIM**.



2. In the BlackBerry PIM Action dialog, select from these configuration options:

Option	Description
BlackBerry PIM Application	Select the BlackBerry PIM application with which the action will interact.
Read from Application	<p>By default, this option is selected, and indicates that the action should retrieve information from the BlackBerry PIM application. When selected, the PIM application is launched and a custom menu called "Import to <client name>" is added to all the PIM applications. This menu appears on the context menu of each PIM application on the device.</p> <p>When the PIM action is called, the specified PIM application is launched. The user selects a specific contact, calendar, memo, or task and clicks the "Import to <client name>" menu item, which activates the PIM action on the Device Application Designer, where it loops through the list of PIM controls on the current screen. PIM controls that have valid and compatible PIM fields and attributes are populated with the corresponding value from the PIM application in which the menu was clicked.</p>

Option	Description
Write to Application	Select this option for the action to save information to the BlackBerry PIM application. Values are taken from the PIM controls, table, or list detail layout. The logical types are then checked to see which PIM fields and attributes are valid. The information is then saved into the corresponding PIM application.
Data Source	When writing information to the BlackBerry PIM application, this option indicates where the client should get information from. For example, if you select Display, the client searches the entire screen display for controls that have logical datatypes mapped. If you select Table, the client searches only in the table. You can select Data Source only if Write to Application is also selected.
Launch PIM Application	This option is available only if Write to Application is selected. If selected, the corresponding application is launched after writing the PIM action.



3. Click **OK**.
4. Select **File > Save**.

Generating a Windows Mobile Device Application

Use the Generate Device Application wizard to generate a Windows Mobile Device Application, launch the Windows Mobile emulator, and run the Windows Mobile device application on the emulator.

Prerequisites

You must have Visual Studio installed.

1. Click the Verify icon  on the toolbar to verify the device application has no errors.
2. Click the code generation icon  on the toolbar.
3. In the Generate Device Application wizard, select **Windows Mobile** and click **Next**.
4. Enter the information for the code generation options:

Option	Description
Favorite Configurations	Select a configuration.

Option	Description
Device	<ul style="list-style-type: none"> • Target device – select the device. • Library version – choose the Microsoft .NET version of the library used to compile the generated code. • Deploy to an ActiveSync connected device or emulator – select this option to deploy the generated code to a Windows Mobile device or emulator. ActiveSync enables the transferring and installation of the application on the mobile device. <hr/> <p>Note: On Windows Vista, ActiveSync has been replaced with Windows Mobile Device Center.</p>
Code Generation	<ul style="list-style-type: none"> • Visual Studio solutions folder – accept the default or click Browse to enter the location for the Visual Studio Solutions folder. • Solution name – enter the name of the Visual Studio solution. • Client project name – enter the name of the project that contains the user interface.

Option	Description
Advanced	<p>When you generate the device application, two projects are generated—the client project, which contains the user interface screens, and the mobile application project, which contains the mobile business objects that are used to access and update the data.</p> <ul style="list-style-type: none"> • Client project namespace – enter the namespace to use for the generated UI classes. • Client project assembly name – the name of the generated .exe file for the project. This is the name that appears on the mobile device. • Mobile application project name – the name of the Mobile Application Project that contains the mobile business objects used in the device application. • Mobile application project namespace – accept the default or enter the name for the mobile application project. • Mobile application project assembly name – accept the default or enter the name for the .dll of the mobile application project. • Client project icon – click Browse to select an icon with which to associate the generated .exe file. This is the icon that appears on the mobile device. • Deployment timeout (minutes) – the maximum time to wait for deployment to the device. • Silent install – use this option only when you are deploying to the emulator (not the device itself). This enables deployment to proceed with no user input. • Generate metadata access classes – select to generate additional metadata classes that describe the attributes and parameters of the generated MBO classes. • Generate base classes in a DLL – select so the base classes are generated in a .dll file, which means you can view the source code, but you cannot edit it. If you do not select this option, base classes are generated as .cs files, along with the other generated .cs files, which can be edited.

Option	Description
	<ul style="list-style-type: none"> • Delete solution folder prior to generation – remove existing source folders before re-generating the Visual Studio solution.

5. Click **Finish**.

Rebuilding the Generated Solution in Visual Studio

After generating code for a Mobile Windows device, you can modify the code in Visual Studio.

Prerequisites

Visual Studio must be installed.

When you generate the code for a Windows Mobile device using the Device Application Designer, the Visual Studio solution is saved to the folder you designated in the Code Generation section of the Generate Mobile Windows Application wizard.

1. In Visual Studio, select **File > Open > Project/Solution**.
2. Browse to the solution file (.sln) you want to open and double-click the file.
3. In Solution Explorer, right-click the solution and select **Rebuild Solution**.
4. Select **File > Save**.
You can now open the form for which you want to modify the code.

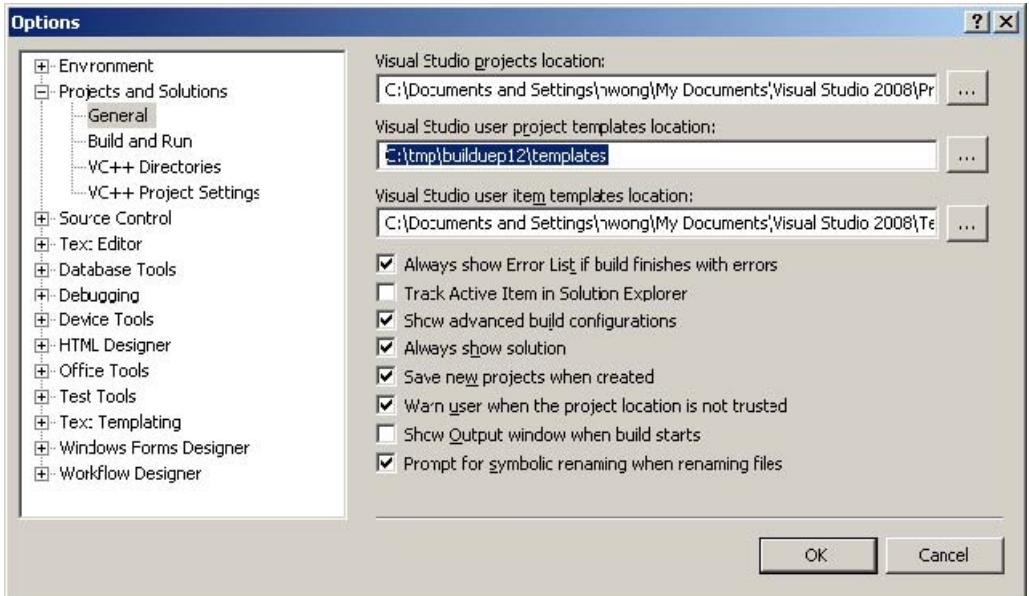
Developing Device Applications for Win32 .NET Platforms

Build a device application that runs on Win32 .NET platforms.

Prerequisites

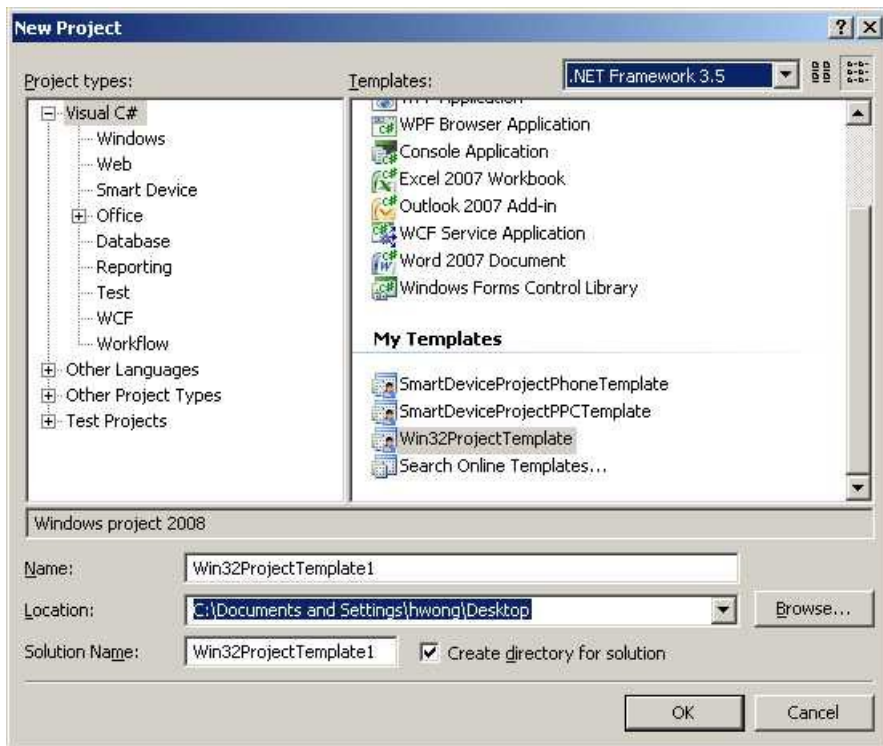
If you are using a File mobile business object (MBO), Afaria must be installed. See the *Sybase Unwired Platform Installation Guide*.

1. Verify that the Projects and Solutions General preferences are set for the template location:



This location determines the templates that appear, and that you can select from when you create a new project.

2. Select **File > New > Project**.



3. From Templates, select **Win32ProjectTemplate**.
4. Develop the mobile business objects (MBOs) that implement the business logic.

See these online help topics:

- *Sybase Unwired Platform 1.2 > Sybase Unwired Workspace 1.2 – Visual Studio Edition > Develop > Developing a Mobile Business Object*
- *Sybase Unwired Platform 1.2 > Sybase Unwired Workspace 1.2 – Eclipse Edition > Develop > Developing a Mobile Business Object*

Note: If you build the mobile business objects in Eclipse, follow the procedure in the help topic *Sybase Unwired Platform 1.2 > Sybase Unwired Workspace 1.2 – Visual Studio Edition > Develop > Developing a Mobile Business Object > Creating a Mobile Application Project > Importing an Eclipse Project*, then build the code for the device application in Visual Studio.

5. (Optional) If you are using a File MBO, select **Add > Existing Item** to add these files as items in the Visual Studio project:
 - `Interop.XeClientLib.dll`
 - `Sybase.UnwiredPlatform.Data.Afaria.dll`
6. Generate the Win32 client code for the mobile business object.
7. Add the generated code to the new project you created from the template.

For more information, see the *Sybase Unwired Platform 1.2 Client Object API Cookbook*.

Correction to Writing a Custom Result Set Filter

The implementation class names that are documented in the topic "Writing a Custom Result Set Filter" are incorrect.

The Sybase Unwired Platform 1.2 topic "Writing a Custom Result Set Filter" incorrectly refers to these class names:

- `com.sybase.eis.ResultSetFilter`
- `com.sybase.eis.ResultSetFilterMetaData`

The correct class names are:

- `com.sybase.uep.eis.ResultSetFilter`
- `com.sybase.uep.eis.ResultSetFilterMetaData`

Troubleshooting

This troubleshooting information is new for Sybase Unwired Platform version 1.2.1.

Restrictions for Configuring Unwired Server Properties

Configure Unwired Server properties only from the Administration Console or by editing the `sup.properties` file.

Problem: Do not modify Unwired Server settings by editing the `configure-sup.xml` file. Doing so can cause unforeseen problems, such as losing configuration information after upgrading.

Solution: Perform configuration tasks directly from the Sybase Control Center (SCC) Administration Console whenever possible, or by editing the `sup.properties` file, if the configuration property is not available from SCC.

For example, change the default synchronization protocol used by Unwired Server from HTTP to HTTPS by modifying the `sup.sync.protocol` entry in the `sup.properties` file to `sup.sync.protocol=https`.

If you modify the `sup.properties` file directly, you must run the command **updateProps.bat -r** for your changes to take effect.

The **-r** flag indicates the values from the `sup.properties` file are applied to `clusterdb` (which is the actual location from which Unwired Server values are retrieved, not directly from `sup.properties`). If you do not run **updateProps.bat -r**, before restarting Unwired Server, old property values are retrieved from `clusterdb` and these values overwrite the changes you made to `sup.properties`.

Make a back-up copy of `sup.properties` before making any changes to it.

Argument and Column Name Length Limitations

Mobile business object (MBO) argument and column names have no character length limits.

While there is a 100 character maximum for MBO parameter and attribute names, there is no such limitation for argument or column names.

Cannot Preview Data

Problem: In some cases, Unwired WorkSpace cannot extract the metadata that is required for a preview through the JDBC driver. In these cases, Unwired WorkSpace either executes the mobile business object's SQL statement to extract the metadata from the preview results, or executes a remote procedure call to get the results.

Solution: In `UnwiredWorkSpace.bat`, set the `autocommitPreviewTransaction` property to true:

1. Shut down Unwired WorkSpace.
2. Go to the Unwired WorkSpace Eclipse subdirectory; for example: `C:\Sybase\UnwiredPlatform-1_2\Eclipse`.
3. Use a text editor to open `UnwiredWorkSpace.bat`, and add this line, following **vmargs**:

```
-D "autocommitPreviewTransaction=true"
```

For example:

```
start "Sybase Unwired WorkSpace" "%ECLIPSE_ROOT%\eclipse.exe"
%ADDITIONAL_ARGS% -vm "%JAVA_HOME%\bin\javaw.exe" -vmargs
-D "autocommitPreviewTransaction=true"
-D"java.endorsed.dirs=%ECLIPSE_ROOT%\endorsed" -Xmx512M -
DSampleObject=true
-DPureJava=PureJava -Declipse.product=com.sybase.sup.SUPproduct
...
goto
EXIT
```

4. Restart Unwired WorkSpace.

Troubleshooting Mobile Business Object Web Service Deployment Failures

Problem: If a Web service contains a parent and a child hierarchical data structure, and the parent and child have columns that have the same name, the mobile business object using this interface cannot be deployed on the server. An error message displays during deployment.

Solution: Avoid using a Web service that contains a parent and child that have columns with the same names.

Default Values are not Recognized as Synchronization Values

Set the sync parameter value before every synchronization from the device application if the default value is not recognized as a sync value.

Problem: A deployed MBO contains a parameter with **Filter by** unselected, and the default or personalization value is not NULL, and the MBO definition contains an operator that is not equal to the user defined default value for the parameter, as this code segment illustrates:

```
"id > @OP["id"]=""]"
```

If you do not set a sync parameter value from the device application, and instead expect Unwired WorkSpace to use the default value as the sync parameter value, the default value is not recognized as a sync parameter.

Solution: Set the sync parameter value before every synchronization from the device application in this scenario.

SQLE_TOO_MANY_PUBLICATIONS Error

Problem: If there are more than 30 syncable mobile business objects (MBOs) in a package, and fewer than 30 syncable MBOs in the Device Application Designer file, a SQLE_TOO_MANY_PUBLICATIONS error may occur when you synchronize the MBOs on Windows Mobile.

Solution:

1. In Unwired WorkSpace, open `Program.cs`, which is located in `%Visual Studio solution folder%\%Client project name%`.
2. Set the value of `Sybase.UnwiredPlatform.Data.DatabaseUtilities.UseDynamicPublication` to true.
3. Rebuild and redeploy the project to your Windows Mobile device.

Device Application Designer Does Not Generate GUI Fields

Problem: The Device Application Designer does not generate graphical user interface (GUI) fields for enterprise information system (EIS) connection property parameters in device applications (especially for SAP applications).

Without the generated fields in the device application GUI, the mobile client user cannot configure those parameters.

Solution: Edit the mobile business object (MBO) in the development environment by setting personalization keys for the connection properties. During runtime, configure the personalization key values.

Trailing Space Causes Synchronization Failure

Problem: If you attempt to synchronize a mobile business object (MBO), and its key column includes values that differ only by a space; for example, "test" versus "test ", synchronization fails.

The Enterprise Information System interprets "test" and "test " as different values, and two rows are created. When you synchronize the MBO, the trailing space is stripped from the value "test ", which creates a conflict, and generates an error.

Solution: Do not insert key values that differ only by a space.

BlackBerry Devices Display a Maximum of 6000 Records

Problem: BlackBerry devices encounter out-of-memory errors when trying to display more than 6000 records (rows) of data.

Solution: Design your mobile business objects (MBOs) and device applications to restrict the number of rows displayed to less than 6000.

Troubleshooting Windows Mobile Device Applications

Problem: When trying to open a device application on a Windows Mobile device, a Cannot instantiate Ctl_image.FormUntitled1(*application name*) [Can not instantiate Form of type FormUntitled1] error is received.

Solution: This error is received if the device application references an image that is larger than 5mb. Do not reference images that are larger than 5mb in size in device applications for the Windows Mobile platform.

Synchronizing Device Applications that Reference Related Mobile Business Objects

Problem: If you synchronize from a child screen of a device application, and if the parent data is missing, old data still appears in the child screen. (Parent data may be missing either because back-end data has been deleted, or because of a changed synchronization condition, such as modified personalization keys.)

Solution: Always synchronize data on the top-level (parent) MBO screen.

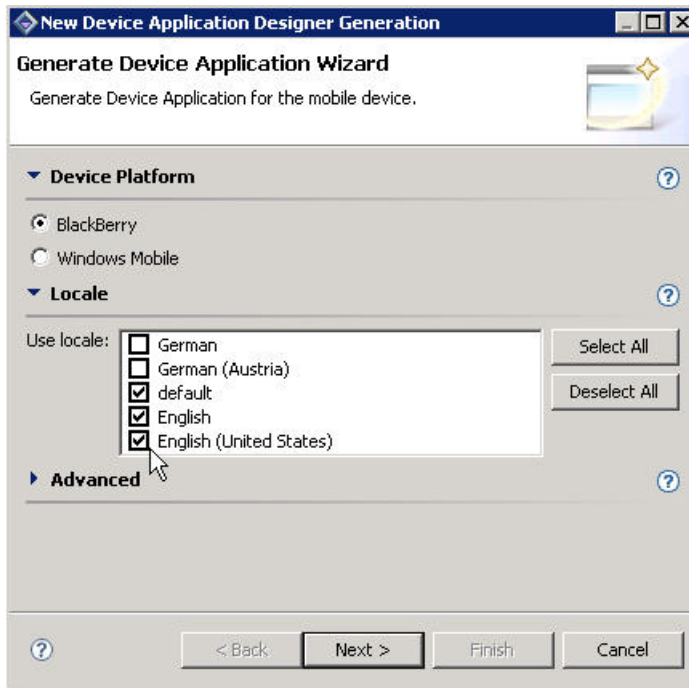
Generation Failed Error When Generating a Device Application

Problem: When generating a device application that uses localization in the Device Application Designer, a "Generation Failed: The locale that contains the basic locale of locale must be included " error is received.

Solution:

- When you specify a country for the language, the basic language locale must also be available. For example, if you create a locale and specify English as the language, then there must also be a locale for English (the basic language).

- If you create a locale that specifies language, country, and variant, the locale for the basic language and the locale for the basic language and the country must be available. For example, if you create a locale and specify English as the language, United States as the country, and WIN as the variant, then English (United States) and English locales must also be available.



When you select a locale with language and country specified, the basic language is also automatically selected. If the basic language locale is not available, you receive the error message.

Missing Sync Parameter Error Message on the BlackBerry Simulator

Problem: After generating the device application code in the Device Application Designer, launching the BlackBerry Simulator, and synchronizing the device application, a "Missing Sync Parameter" error appears.

Solution: The server does not save the default value for sync-only parameters during deployment. The behavior for sync-only parameters is:

1. When a personalization key is defined for sync-only parameters, the default value is ignored.
2. When a personalization key is not defined *and* a default value is defined, then the default value is used if you select the Use server default value for initial

synchronization when you create the device application in the Device Application Designer.

The other situation in which this error is received is when there is a relationship defined for the MBO you are syncing. If you synchronize the second level MBO from the synchronization screen, the second MBO does not get the value that is defined for the relationship parameter.

New Features - Visual Studio Edition

New features, updated documentation, and new troubleshooting topics are described.

The tables below briefly describe each feature, and provide links to associated topics.

New Sybase Unwired Platform features

Feature	Topics
For SAP mobile business objects, you can add a predefined or customized result checker.	<i>Adding an SAP Result Checker on page 71</i>
Optimize device application and Unwired Server performance by defining a cache-update policy when you develop mobile business objects.	<i>Cache Update Policy</i>
If the SAP AutoCommit feature is enabled, a successful operation is always committed.	<i>Configuring the SAP Auto-Commit Feature on page 85</i>
Linked parameters use the dynamic format of mobile business objects to determine the content in device applications.	<i>Creating a Data Source for Linked Parameters</i>
Multilevel (chained) insert operations allow you to synchronize multiple Web-service mobile business objects that have a defined relationship.	<i>Creating Multilevel Insert Operations for Web Service Mobile Business Objects</i>
You can modify SAP Java connector (JCo) properties, and use the property values as parameters.	<i>Modifying SAP Connection Properties on page 98</i>

Documentation updates

Feature	Topics
Corrected information in steps 1 and 2. In step 1, the directory name is BIN32, not win32. In step 2, call createcert , not gencert .	<i>Generating Certificates to Enable HTTPS Synchronization</i>

Feature	Topics
Replaced the topic "Configuring Your Environment for SAP" with the topic "Preparing the Unwired Platform Environment for SAP Connections."	<i>Preparing the Unwired Platform Environment for SAP Connections</i>

Troubleshooting

Problem	Topic
If a mobile business object has more than one operation of type "Other," the generated code GetCalledOperations returns only one operation.	<i>Codegen.GetCalledOperations Returns Only One Operation</i> on page 101
If the sum of the column sizes for attributes and parameters exceeds 64K, the default page size of the device database is too small.	<i>Device Database Page Size</i> on page 101
If you cannot connect to SAP from the Visual Studio development environment, and an Open file C:\WINDOWS\sapmsg.ini failed message appears, add the missing sapmsg.ini file to your environment.	<i>SAP Connection Error</i> on page 102
When you use the Detailed Properties dialog to edit SAP operation parameters, parameter-to-argument linking may be lost.	<i>SAP Parameter-to-Argument Linking</i> on page 102
If you choose to let the system generate the user interface when you generate client code for a project, it may take five minutes or more.	<i>Slow Code Generation</i> on page 102
If there are more than 30 MBOs in a package, and less than 30 MBOs in the Device Application Designer file, a SQLE_TOO_MANY_PUBLICATIONS error may occur when you synchronize the MBOs on Windows Mobile.	<i>SQLE_TOO_MANY_PUBLICATIONS Error</i>
If you attempt to synchronize a mobile business object, and there are two rows in the EIS with key values that differ only by a trailing space, synchronization fails.	<i>Trailing Space Causes Synchronization Failure</i>
If you are using the Chinese Edition of Visual Studio 2008, you must manually move some files for the installation to work with Sybase Unwired Platform.	<i>Visual Studio 2008 Chinese Edition</i> on page 103

New Features

These new features are included in Sybase® Unwired Platform version 1.2.1.

Adding an SAP Result Checker

Add or create an SAP result checker for a mobile business object. You can add a predefined result checker, or create a new one.

1. In the Mobile Business Object Properties dialog, select the **Definition** tab.
2. For the **SAP Result Checker**, select one of these options:

Option	Description
Default	If there is a RETURN parameter in the currently selected BAPI operation, this option is automatically selected. This option uses the <code>com.sybase.sap.DefaultSAPResultCheck</code> result checker.
None	If there is not a RETURN parameter in the currently selected BAPI operation, this option is automatically selected. This option uses the <code>com.sybase.sap.NoOpSAPResultCheck</code> result checker.
Custom	To specify a custom SAP result checker, select this option, and enter the Java class name .

3. Implement the new class by writing the implementation on top of the skeleton, as documented in the topic, *Writing a Custom SAP Result Checker*.
4. Test and preview the results of your result checker:
 - a) To reuse input values you have previously saved, select **Existing Configuration**. Otherwise, load the defaults, or create a new set of input values expressly for this preview instance.
 - b) Click **Preview**.

If the data runs successfully, `Execution Succeeded` appears in the dialog, and data appears in the **Preview Results** window.

Writing a Custom SAP Result Checker

An SAP result checker is a custom Java class that implements error checking for SAP mobile business objects.

Since not all SAP BAPIs or RFCs use the "standard" error reporting technique, you can implement your own custom SAP result checker. This allows you to check any field for errors, or implement logic that determines what constitutes an error, and the severity of the error.

1. Provide a Java class that implements the `SAPResultChecker` interface:

```
package com.sybase.sup.sap;
public interface SAPResultChecker
{
    /**
     *
     * @param f - JCO function that has already been executed.
     * Use the JCO API to retrieve returned values and determine if
     the RFC has executed
     */
}
```

New Features 1.2.1

```
* successfully.  
* @return a single Map.Entry. The boolean "key" value should  
be set to true if the  
* RFC is deemed to have succeeded. Normal result processing  
will ensue.<P>  
* If the String value is not empty/null, that value will be  
treated as a warning message,  
* which will be logged on the server,  
* and returned as a warning in transaction logs to the  
client.<P>  
* Set the key value to false if it is deemed the RFC has  
failed. The String value will  
* be thrown in the body of an exception. The error will be  
logged on the server, and the  
* client will receive a transaction log indicating failure,  
including the string value.  
*/  
    Map.Entry<Boolean, String> checkReturn(JCO.Function f);  
}
```

There are two `SAPResultChecker` implementations located in `%SUP_HOME%\Servers\UnwiredServer\uep\tomcat\webapps\onepage\samples\sap` directory, which you can modify and reuse as custom result checkers:

- `DefaultSAPResultCheck` – the default SAP result checker.
 - `NoOpSAPResultCheck` – this result checker always returns a successful result.
2. Save any classes you create to an accessible Unwired Workspace location. This allows you to select the class when you configure the SAP result checker for your mobile business object.

Deploying SAP Result Checker Classes to Unwired Server

Before deploying SAP mobile business objects that use result checker classes, copy the compiled classes to `<Unwired-Server-install>\lib\filters\<packageName>` on the primary Unwired Server machine.

You can also place result checker classes into a directory structure; the root of which should start at `packageName`. During cluster synchronization, the SAP result checker classes are automatically distributed to other Unwired Servers in the cluster.

1. (Optional) To maintain and deploy a single file, create a Java archive of all your classes.
2. Choose the Unwired Server target location for the result checker classes, dependencies, and third-party JARs:
 - To avoid restarting Unwired Server, copy either the JAR file or the individual class files to:
`<Unwired-Server-install>\lib\filters\<DeploymentPackageName>`.
For example, if you have a `ResultChecker_1.0.0` package and a `com.acme.filters.ResultChecker` result checker class, create

```
<Unwired-Server-install>\lib\filters
\ResultChecker_1.0.0\com\acme\filters
\ResultChecker.class. Or, create acmeFilters.jar from the
com.acme.filters.ResultChecker classes, and copy it to <Unwired-
Server-install>\lib\filters
\ResultChecker_1.0.0\acmeFilters.jar.
```

- To restart Unwired Server after deployment, copy the classes to:
<Unwired-Server-install>\lib\filters\.

3. Deploy the SAP mobile business object to Unwired Server.

Configuring an SAP Result Checker

Configure an SAP result checker for a mobile business object.

1. In the Mobile Application Diagram, select **Show Properties View**.
2. In the Properties view, click the **Attributes** or **Operations** tab, then the **Definition** tab.
3. Click **Edit**.
4. Make your changes in the Change Definition dialog, and click **OK**.

Refactoring an SAP Result Checker

When an SAP result checker is deleted, renamed, or moved, update its references automatically.

Deleting References to an SAP Result Checker

Delete all references to an SAP result checker from the workspace.

1. Open the mobile application project that contains the SAP result checker you want to delete, and expand the **Filters** folder.
2. Right-click the SAP result checker and select **Refactor > Delete**.
3. In the Confirm Delete dialog, verify the selected references, and click **OK**.

Moving an SAP Result Checker

Move an SAP result checker to another location, and update its references in the workspace.

1. Open the mobile application project that contains the SAP result checker you want to move, and expand the **Filters** folder.
2. Right-click the SAP result checker, and select **Refactor > Move**.
3. In the Move dialog, confirm the changes, and click **OK**.

Renaming an SAP Result Checker

Rename an SAP result checker, and update its references in the workspace.

1. Open the mobile application project that contains the SAP result checker you want to rename, and expand the **Filters** folder.
2. Right-click the SAP result checker, and select **Refactor > Rename**.

3. In the Rename Type dialog, verify the changes, and click **Finish**.

Searching for References to an SAP Result Checker

Search for mobile business objects that reference SAP result checker classes.

SAP mobile business objects can share SAP result checker classes, within a single project or across multiple projects.

1. In WorkSpace Navigator, in the mobile application project, expand the **Filters** folder.
2. Right-click the SAP result checker, and from the context menu, select **References > Mobile Business Object**.

The search results appear in the **Search** pane.

Cache Update Policy

Fine-tune device application and Unwired Server performance by defining a cache update policy for mobile business object operations.

Setting a cache update policy for mobile business object (MBO) operations gives you more control of both Unwired Server interactions with the enterprise information system (EIS) to which the MBO is bound, and consolidated database updates. Fine-tuning these interactions and updates improves both Unwired Server and device application performance.

Note: Consolidated database, CDB, and cache all refer to the same thing, and the terms are used interchangeably.

- MBO operations perform specific functions based on their definition:
 - Primary read operation – the EIS operation used to define and initially populate the CDB (from the EIS) for the MBO.
 - Create, update, delete (CUD operations) – modify EIS data depending on the definition of the operation. Unwired Server maintains a cache (CDB) of back-end EIS data to provide differential synchronization and to minimize EIS interaction. When an operation is submitted from a device application to the EIS, the cache must be refreshed.

While these types of bulk-fetch and CDB caching are effective in reducing the number of interactions required with the back-end EIS, and work well in some other cases (where MBO data is occasionally updated in the back-end), performance suffers if changes are initiated from Unwired Server (by way of MBO operations), or if changes are frequent.

The cache update policy introduces alternative methods of updating the cache at finer granularity, which improves performance.

- Alternate read operations – can be invoked either from:
 - A chained read cache policy to augment CUD operations by chaining an alternate read operation to a CUD operation.

- A data change notification, which provides a mechanism to invoke MBO operations, including alternate read operations. This mechanism is independent of a cache update policy.
- Cache update policy – determines how the CDB is updated after an operation. You can set the cache update policy for operations, with these exceptions:
 - Operations defined as "Other" do not support alternate read or a cache update policy.
 - When invoked, alternate read operations always use the apply operation results cache update policy.

Versions of Sybase Unwired Platform earlier than 1.2.1 supported only the invalidate cache policy—any CUD or other operation issued from a device application invalidated the cache and required a primary read operation to refresh the cache.

In Unwired Platform version 1.2.1, these are the five cache update policies you can associate with MBO CUD operations:

- Invalidate cache
- No invalidate cache
- Apply operation result
- Apply operation parameters
- Chained read

When an MBO CUD operation is called, its cache update policy determines how operation results are applied to the consolidated database. Generally, there are two ways of calling an MBO operation:

1. Device client calls the operation.
2. A data change notification (DCN) request contains the operation.

Note: Other methods used to update the CDBs that are external to MBO operations, and not associated with cache update policies include:

1. EIS-initiated DCN – an HTTP request to Unwired Server, in which the DCN request contains the payload (information about the changed data).

Note: EIS-initiated DCN also supports HTTP POST requests which provides a higher level of security.

2. Scheduled data refresh – defined in Sybase Control Center; polls the EIS for changes at specified intervals.
-

Setting the Cache Update Policy

The cache update policy defines how the consolidated database is updated after calling a mobile business object operation at a finer granularity than the primary read operation.

To maximize Unwired Platform efficiency and performance, set the cache update policy, which determines how create, update, and delete operation results from the Enterprise Information System (EIS) are applied to the consolidated database (CDB).

Note: These terms all represent the same thing: consolidated database, CDB, and cache.

1. In the Mobile Application Diagram, double-click the operation.
2. In the Operation Properties dialog, select the **Cache Policy** tab.
3. Select the cache update policy:

Cache Update Policy	Description
Invalidate	Invalidates and refreshes the CDB after the client calls the mobile business object (MBO) operation. This is the default cache update policy.
No invalidate	The CDB remains unchanged after the client calls the MBO operation.
Apply operation result	Updates the CDB based on the result set that is returned from the MBO operation.
Apply operation parameters	Updates the CDB based on the operation's parameters.
Chained operation	<p>Chain an alternate read operation to the MBO operation. The CDB is updated with the results returned by the alternate read operation. If you select this option, a Chained Operation drop-down list appears, which allows you to either choose an existing operation, or create a new one by selecting New. Define the chained operation the same as other operations; set the operation type to "Read."</p> <hr/> <p>Note: Find by must be selected for at least one attribute.</p> <hr/> <p>The alternate read operation is no different from an operation that uses the apply operation result cache update policy. Specifying a different read type operation enables chaining it to the MBO operations to get the desired results.</p>

Apply Operation Result Cache Policy

Use the apply operation result cache policy to update the consolidated database, based on the results returned by the operation..

The record set that is returned from the operation is processed and applied to the consolidated database.

This example uses the "customer" table, which is in the sampledb database:

1. Drag and drop the "customer" table onto the Mobile Application Diagram to create a mobile business object.
2. Double-click the **Create** operation to open the Operation Properties dialog.
3. Select the **Definition** tab, and update the SQL query to:

```
INSERT INTO sampledb.dba.customer
(id,
fname,
lname,
address,
city,
state,
```



```

zip,
phone,
company_name)
VALUES
(@OP["id"= " "],
'@OP["fname"= " "]',
'@OP["lname"= " "]',
'@OP["address"= " "]',
'@OP["city"= " "]',
'@OP["state"= " "]',
'@OP["zip"= " "]',
'@OP["phone"= " "]',
'@OP["company_name"= " "]'
)
SELECT * FROM customer where id = @OP["id"= " "]

```

4. Select the **Cache Policy** tab, and select **Apply operation result**.
5. Click **Apply**.
6. CDb and client data updates are based on the operation definition and the apply operation result policy; other changes to the Enterprise Information System are ignored.

Apply Operation Parameters Cache Policy

Use the apply operation parameters cache policy to apply the operation's parameters to the consolidated database.

The values of the operation parameters are directly applied to the consolidated database (CDB).

Note: These terms all represent the same thing: consolidated database, CDb, and cache.

This example uses the department table, in the sampledb database:

1. Drag and drop the department table onto the Mobile Application Diagram to create a mobile business object (MBO).
2. In the MBO, double-click the **Update** operation. The Operation Properties dialog opens.
3. Select the **Cache Policy** tab, and select **Apply operation parameters** for the **Cache Update Policy**.
4. Deploy the MBO to Unwired Server.
5. In Sybase Control Center, set the MBO's **Cache Interval** to one hour (or any value sufficiently long to complete this test).
6. Synchronize the MBO from a test client or device application. The client, the CDb, and the EIS all have the same data:

dept_id	dept_name	dept_head_id
100	R & D	501
200	Sales	902
300	Finance	1293
400	Marketing	1576
500	Shipping	703

7. Call the MBO's update operation from the test client or device application to update this record:

```
dept_id=400, dept_name="QA", dept_head_id=1576
```

The EIS is modified:

dept_id	dept_name	dept_head_id
100	R & D	501
200	Sales	902
300	Finance	1293
400	QA	1576
500	Shipping	703

8. Synchronize the MBO. Based on the apply operation parameters policy, which is associated with this operation:

1. The dept_id=400 record is updated using the update operation's parameter values (an inferred read), and becomes:

```
dept_id=400, dept_name=QA, dept_head_id=1576
```

2. The CDb and the client are updated, and contain:

dept_id	dept_name	dept_head_id
100	R & D	501
200	Sales	902
300	Finance	1293
400	QA	1576
500	Shipping	703

Chained Operation Cache Policy

The chained operation cache update policy allows you to chain a read operation to create, update, or delete operations, and update the consolidated database (CDB) at a finer granularity.

The chained operation cache policy allows chaining of an alternate read operation to a create, update, or delete (CUD) operation. The parameters of the alternate read operation should be a subset of the parameters of the chaining operation. The parameter values of the chaining operation are passed to the alternate read operation and the results returned from the alternate read operation are then applied to the CDB.

Note: These terms all represent the same thing: consolidated database, CDB, and cache.

This example uses the department table, which is in the sampledb database:

1. Drag and drop the department table onto the Mobile Application Diagram to create a mobile business object (MBO).
2. In the MBO, double-click the **Create** operation to open the Operation Properties dialog.
3. Select the **Cache Policy** tab, and select **Chained operation**.
4. Select **New** to launch the Mobile Business Operation Creation wizard, and define the chained operation:
 - Name – ChainedRead.
 - Operation type – Read.
 - Connection type – Database.

- Connection name – select a connection to the sampled database.
- Define the SQL query – `select * from department where dept_id < 400`.
- Click **Finish**.

5. Deploy the MBO to Unwired Server.

The result is an MBO that when accessed from a device application or test client behaves as follows (assuming the MBO's **Cache interval** is sufficiently long to complete the test without requiring a cache refresh):

1. The create operation inserts records into the sampled database.
2. The read operation is called after the create operation, and updates the CDB with records where `dept_id < 400`.

From an end-to-end perspective:

1. Initially, synchronizing this MBO from a test client or a device application results in the client, CDB, and EIS with the same data:

```
dept_id dept_name dept_head_id
-----
100      R & D      501
200      Sales      902
300      Finance    1293
400      Marketing  1576
500      Shipping   703
```

2. This record is inserted into the EIS (using some method other than the MBO operation):

```
dept_id=1000, dept_name="QA", dept_head_id=501
```

The EIS now contains:

```
dept_id dept_name dept_head_id
-----
100      R & D      501
200      Sales      902
300      Finance    1293
400      Marketing  1576
500      Shipping   703
1000     QA         501
```

Neither the client nor the CDB are updated with the new record.

3. The client invokes the create operation to insert the record:

```
dept_id=350, dept_name="QA", dept_head_id=501
```

4. The chained read operation is invoked.

5. The client synchronizes the MBO. While the EIS contains `dept_id=1000` and `dept_id=350`:

```
dept_id dept_name dept_head_id
-----
100      R & D      501
200      Sales      902
300      Finance    1293
```

New Features 1.2.1

400	Marketing	1576
500	Shipping	703
1000	QA	501
350	QA	501

Based on the read operation's definition, the only new record retrieved from the EIS and updated in the CDB and the client is the `dept_id=350` record:

dept_id	dept_name	dept_head_id
100	R & D	501
200	Sales	902
300	Finance	1293
400	Marketing	1576
500	Shipping	703
350	QA	501

Defining Chained Operations

A chained operation acts upon the results from a previous operation, and applies the results to the consolidated database (CDB) using the apply operation results policy.

Primary read is the EIS operation for an MBO that defines how it is populated. A new operation of type "read" is introduced whose cache update policy is always apply operation result. It returns a record set that is applied to the CDB. These kind of operations are called as chained read operations. There can be any number of chained read operations defined on an MBO. Chained read operations generally return data corresponding to a finer granularity than the primary read of the MBO. This enables updating the cache at finer granularity. The chained read operations can be chained to any create, update, delete (CUD) operations of the MBO to achieve the desired results. The Device API does not support calling chained read operations directly, but can be invoked independently using DCN request.

Note: These terms all represent the same thing: consolidated database, CDB, and cache.

1. In the Mobile Application Diagram, double-click the operation to which you want to chain the operation.
2. In the Operation Properties dialog, select the **Cache Policy** tab.
3. For **Cache Update Policy**, select **Chained operation**.
4. For the **Chained Operation**, select **New**.
5. In the Mobile Business Operation Creation wizard, enter a name for the operation, and select **Read** as the operation type.
6. Select the connection type, and either select the connection name, or click **New**, and create a new connection. Click **Next**.
7. Complete the operation definition according to the data source type to which you are binding the operation. For example, for a Web-service data source, from the XSLT Definition screen, select **Configure XSLT** to access the XSLT.

8. Modify the operation to meet the intended need. For example, you may have an MBO operation that inserts records into the database. The chained operation filters the results, so that only those for a particular user are inserted into the CDB.

Chained Operation Requirements

Mobile business objects must meet certain requirements before you can add chained operations to them.

To add a chained operation to a mobile business object (MBO), the MBO must meet these requirements:

- Be bound to a data source that has a primary key, and has one or more Find by attributes set.
- The record set returned by the chained read operation must have all columns mapped to key attributes (Find by attributes). You can map these either when you create the chained operation, or by editing the operation in the Properties view.
- The result set of the chained operation must contain all MBO playback and sync parameters (Filter by parameters and attributes).

Playback and Synchronization Parameters

The result set of a chained operation must contain the playback and synchronization parameters of the operation to which it is chained.

For a chained operation to be valid, its result set must contain the playback and sync parameters of the chaining mobile business object (MBO) operation.

For example, create three MBOs by dragging-and-dropping the "sampledb.dba.customer" data source. For each MBO, set **Filter by** for both the "state" parameter and the "company_name" attribute, so "state" is a playback and sync parameter in all three MBOs:

```
SELECT id, fname, lname, address, city, state, zip, phone,
company_name
FROM sampledb.dba.customer
WHERE state=@OP["state"]="CA"]
```

This simple example illustrates validation errors. Create a chained operation for each MBO using these SQL statements:

- Customer1 MBO

```
SELECT * from customer
```

This is valid, since the result set of the chained operation contains the "state" and "company_name" playback and sync parameters.

- Customer2 MBO

```
SELECT id, fname, lname, phone, company_name
FROM customer
WHERE company_name=@OP["company_name"]="AAA"]
```

This is an error, since the result set does not contain the "state" playback and sync parameter. It contains a different sync parameter, "company_name."

- Customer3 MBO

```
SELECT id, lname, fname, phone, company_name
FROM customer
WHERE state=@OP[ "state"="CA" ]
```

This is an error. Even though the operation's parameter is "state," the result set does not contain the "state" playback and sync parameter. It contains another sync parameter, "company_name."

Invalidate Cache Policy

Use the invalidate cache policy only when other policies cannot be implemented or performance is not an important consideration.

An operation that uses the invalidate cache policy:

1. Performs the create, update, or delete (CUD) operation. For example, insert a new record in the enterprise information system (EIS).
2. Invalidates the cache (CDB) for that mobile business object (MBO) instance.
3. Requires the MBO instance in the CDB to be refreshed from the EIS (this is known as playback). Internally, the MBO's primary read operation executes to retrieve data from the EIS and repopulates the cache.

Invalidate cache is the default cache update policy for any CUD operation for which there is no cache update policy set.

No Invalidate Cache Policy

Use the no invalidate cache policy when changes to the enterprise information system (EIS) do not need to be immediately passed to the consolidated database (CDB).

An operation that uses the no invalidate cache policy updates the EIS without invalidating the cache (CDB) for that instance of the MBO. Because cache is **not** invalidated, it may be different from the EIS. The cache is updated later, for example, based on the **Cache Interval** set on Unwired Server. The no invalidate cache policy eliminates nonessential CDB refreshes, which improves performance.

Note: Consolidated database, CDB, and cache all refer to the same thing, and the terms are used interchangeably.

This example uses the department table from the sampledb database, which is accessible from the "My Sample Database" connection profile.

1. To create a mobile business object (MBO), drag and drop the sampledb's "department" table onto the Mobile Application Diagram.
2. Set the update operation's cache update policy to **No Invalidate Cache**.
3. Deploy the MBO.

4. From Sybase Control Center (SCC) Administration Console, set the MBO's **Cache Interval** to one hour (or any value long enough to complete this test).
5. Sync the MBO from a test client or device application. The client, CDB, and EIS all have the same data:

dept_id	dept_name	dept_head_id
100	R & D	501
200	Sales	902
300	Finance	1293
400	Marketing	1576
500	Shipping	703

6. Call the MBO's update operation from the test client or device application to update this record:

```
dept_id=100, dept_name="SUPQA", dept_head_id=501
```

Sync the MBO. The EIS is modified:

dept_id	dept_name	dept_head_id
100	SUPQA	501
200	Sales	902
300	Finance	1293
400	Marketing	1576
500	Shipping	703

Based on the no invalidate policy associated with this operation, the CDB and client remain unchanged because the cache (CDB) remains valid:

dept_id	dept_name	dept_head_id
100	R & D	501
200	Sales	902
300	Finance	1293
400	Marketing	1576
500	Shipping	703

Cache Update Policy Validation Rules

Validation rules are enforced when you set the cache update policy.

The following error messages may appear when you define a cache update policy:

Severity and message	Meaning	Occurs when you
Must specify a read operation for operation "{0}" when the cache update policy is 'Chained operation'	The type of a chained operation must be "Read."	Create a chained operation

Severity and message	Meaning	Occurs when you
Error: No "Find By" attributes set for the mobile business object, cannot create alternate read type operation	While a primary key for the data source is not necessary (Web-service data sources do not have primary keys), the mobile business object (MBO) must have Find by set for at least one attribute.	Create a chained operation
Error: The columns of the alternate read operation '{0}' must contain the "Find By" attributes of the associated mobile business object	The result-set columns in the chained operation do not contain all Find by attributes.	Create a chained operation
Error: Cache update policy of operation "{0}" is set to "{1}", but there is no attributes with "Find By" set for the mobile business object	The MBO must have at least one Find by attribute.	Set the cache update policy to "apply operation result," "apply operation parameters," or "chained operation"
Error: Operation "{0}": when the cache update policy is "Apply Operation Parameters", the operation parameters' "Fill From Attribute" must have "Find By" set	All of the MBO's key attributes (Find by attributes) must have the Fill from attribute property set to the column of the operation's parameters.	Set the cache update policy to "apply operation parameters"
Error: Must specify an alternate read operation for operation "{0}" when the cache update policy is "{1}"	No chained operation is specified, and the cache update policy is "chained operation."	Set the cache update policy to "chained operation," or delete the chained operation
Error: Parameters of the chained operation "{0}" should be a subset of the parameters of the chaining operation "{1}"	The parameters of the chained operation must be a subset of the parameters of the chaining operation.	Select a chained operation while defining the "chained operation" policy
Error: The columns of the read operation "{0}" must cover 'filter by' parameters of the associated mobile business object"	The result set of a chained operation must contain the playback and synchronization parameters of the mobile business object operation to which it is chained.	Define a chained operation

Severity and message	Meaning	Occurs when you
Warning: Alternate read operation "{0}" must cover all mobile business object attributes with "Filter By" set.	The result set of a chained operation must contain the playback and synchronization parameters of the mobile business object operation to which it is chained.	Define a chained operation

Configuring the SAP AutoCommit Feature

The SAP AutoCommit feature determines whether `commit` is always called after an operation succeeds.

For an SAP operation, if the AutoCommit feature is enabled, `commit` is always called following a successful operation; if the operation fails, changes are rolled back. By default, the AutoCommit feature is enabled; if you disable it, you must explicitly call `commit` after the operation succeeds.

1. In the Mobile Business Object Properties dialog, select the **Definition** tab.
2. To enable the AutoCommit feature, check **Commit SAP Operation**; to disable, uncheck **Commit SAP Operation**.
3. Click **Apply**, and click **OK**.
4. If the SAP operation is bound to a mobile business object (MBO) operation:
 - a) Double-click the MBO operation to open the Operation Properties dialog, and select the **Definition** tab.
 - b) Repeat steps 2 and 3.

Creating a Data Source for Linked Parameters

To implement linked parameters, you must first build a mobile business object that contains all of the dependent values, and a relationship between the values.

The mobile business object can use any data source, for example:

- A database table
- A Web service
- Result-set filter generated

You must convert the data output from these data sources into a table of a specific format by using a transformation method, for example, using an XSLT template for the Web-service output, or using a stored procedure, or a formatted SQL query.

Required attribute names for linked parameters are:

- "index"
- "display_name"
- "value_data"

- "value_name"
- "link"

For example, a correctly formatted SQL table looks like this:

```
CREATE TABLE "dba" . "linked_params" (
  "index" INT NOT NULL ,
  "display_name" VARCHAR (50) NOT NULL ,
  "value_data" VARCHAR (50) NOT NULL ,
  "value_name" VARCHAR (50) NOT NULL ,
  "link" INT NOT NULL ,
)
IN SYSTEM
;
ALTER TABLE "dba" . "linked_params"
  ADD CONSTRAINT "ASA105" PRIMARY KEY CLUSTERED ( "index" )
;
ALTER TABLE "dba" . "linked_params"
  ADD CONSTRAINT "ASA106" UNIQUE NONCLUSTERED ( "index" )
;
```

Linked Parameters

Linked parameters use the dynamic format of the mobile business object to drive the content of the drop-down values on the device application.

Many applications have drop-down lists of values you can choose, and the values you select can affect the choices in other lists.

Linked parameters make it easier to build entry forms for device applications that allow dependent drop-down choices. For example, if a user selects "California" for the state field, the country field is automatically set to "USA." This makes the input data consistent, with less chance of errors. For example:

Field name	Values
Field 1	Accepts values A, B, or C
Field 2	Allows values dependent on Field 1: A->1, 2, 3 B->4, 5, 6 C->7, 8, 9
Field 3	Allows values dependent on Field 1: A->100, 101, 102 B->200, 201, 202 C->300, 301, 302

Field name	Values
Field 4	Allows values dependent on Field 2: 1->11, 111 2->22, 222 3->33, 333 4->44, 444 5->55, 555 6->66, 666 7->77, 777 8->88, 888 9->99, 999

When the user selects value "A" in Field 1, the allowed values listed in Field 2 change to 1, 2, 3; the Field 3 choices become 100, 101, 102.

Note: For each change to Field 2, there is a cascading effect to Field 4.

If the user chooses value C in Field 1, the Field 2 choices become 7, 8, 9; Field 3 choices become 300, 301, 302; Field 4 choices become 77,777, 88,888, 99,999.

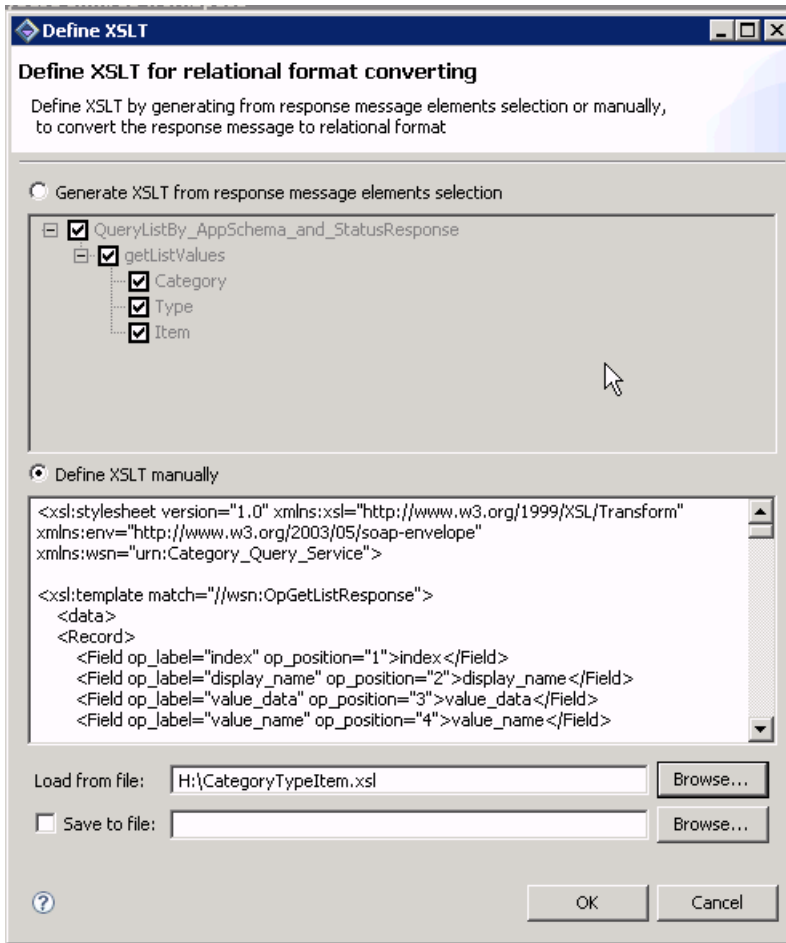
Transforming a Data Source for Linked Parameters

You must transform, or generate, the data source to produce data in the required format for linked parameters.

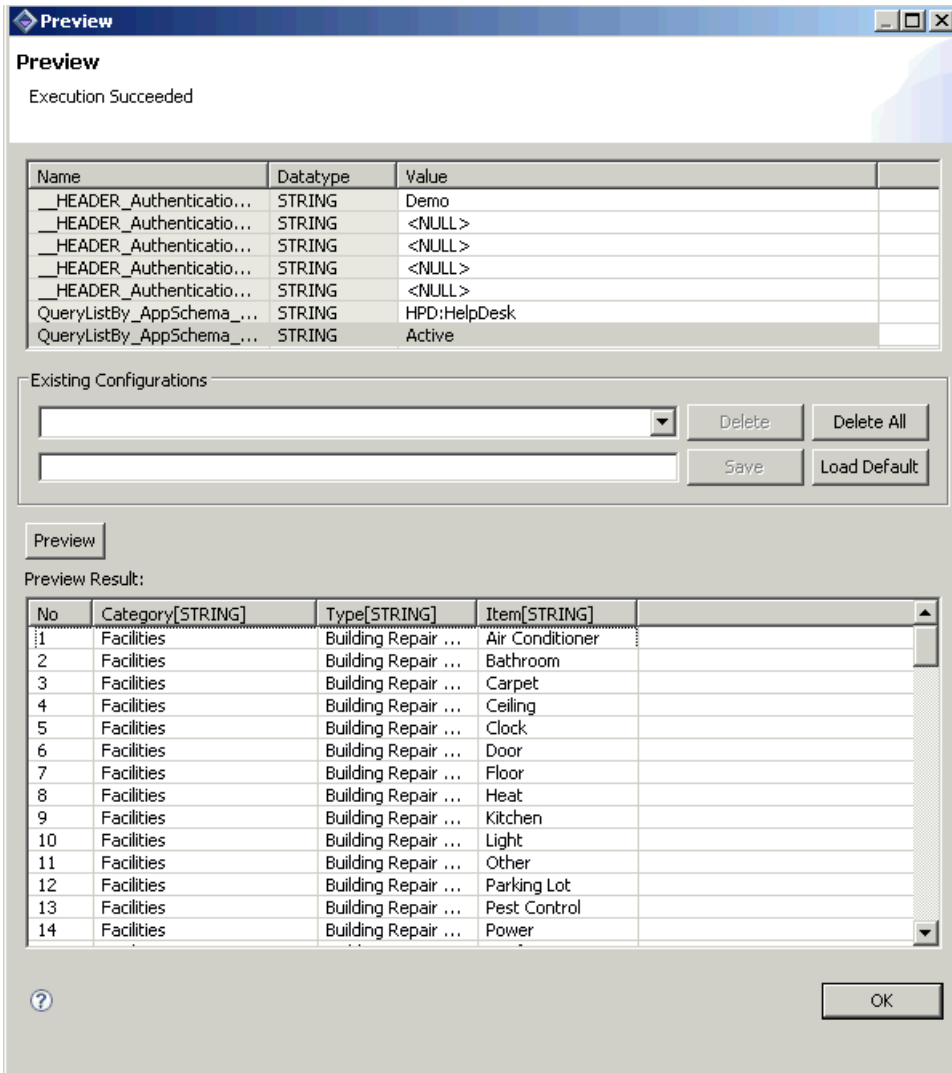
Sybase Unwired Platform includes two XSLT files (`CategoryTypeItem.xsl` and `RegionSiteDepartment.xsl`) located in `<SUP Installation Directory>\UnwiredPlatform\Unwired_Workspace\samples`, which you can modify and use with Remedy Web Services. Use these XSLT files to build mobile business objects to produce the Category-Type-Item or Region-Site-Department dependent field values (linked parameters) that are common in the Remedy HelpDesk and other applications.

If you use a different application or data source, you can use your own transformation or generation techniques to produce the data in a similar format to that required for linked parameters.

This example shows the `CategoryTypeItem` XSLT:



This is a preview of a mobile business object that uses linked parameters:



Example of a database source

This sample creates a linked table using Remedy data.

```
CREATE TABLE "dba"."linked_params" (
  "index" INT NOT NULL,
  "display_name" VARCHAR(50) NOT NULL,
  "value_data" VARCHAR(50) NOT NULL,
  "value_name" VARCHAR(50) NOT NULL,
  "link" INT NULL,
)
IN SYSTEM
;
```

New Features 1.2.1

```
ALTER TABLE "dba"."linked_params"
    ADD CONSTRAINT "ASA105" PRIMARY KEY CLUSTERED ("index")
;
INSERT INTO dba.linked_params
    ("index",
     "display_name",
     "value_data",
     "value_name",
     "link")
VALUES
    (1,
     'Category',
     'default',
     'Default',
     null);
INSERT INTO dba.linked_params
    ("index",
     "display_name",
     "value_data",
     "value_name",
     "link")
VALUES
    (2,
     'Category',
     'hardware',
     'Hardware',
     null);
INSERT INTO dba.linked_params
    ("index",
     "display_name",
     "value_data",
     "value_name",
     "link")
VALUES
    (3,
     'Category',
     'software',
     'Software',
     null);
INSERT INTO dba.linked_params
    ("index",
     "display_name",
     "value_data",
     "value_name",
     "link")
VALUES
    (4,
     'Type',
     'default',
     'Default',
     1);
INSERT INTO dba.linked_params
    ("index",
     "display_name",
     "value_data",
     "value_name",
```

```

        "link" )
VALUES
(5,
 'Type',
 'drive',
 'Drive',
 2);
INSERT INTO dba.linked_params
("index",
 "display_name",
 "value_data",
 "value_name",
 "link" )
VALUES
(6,
 'Type',
 'laptop',
 'Laptop',
 2);
INSERT INTO dba.linked_params
("index",
 "display_name",
 "value_data",
 "value_name",
 "link" )
VALUES
(7,
 'Type',
 'memory',
 'Memory',
 2);
INSERT INTO dba.linked_params
("index",
 "display_name",
 "value_data",
 "value_name",
 "link" )
VALUES
(8,
 'Type',
 'email',
 'Email',
 3);
INSERT INTO dba.linked_params
("index",
 "display_name",
 "value_data",
 "value_name",
 "link" )
VALUES
(9,
 'Type',
 'internet',
 'Internet',
 3);
INSERT INTO dba.linked_params

```

```

("index",
  "display_name",
  "value_data",
  "value_name",
  "link")
VALUES
(10,
 'Item',
 'default',
 'Default',
 4);
INSERT INTO dba.linked_params
("index",
  "display_name",
  "value_data",
  "value_name",
  "link")
VALUES
(11,
 'Item',
 'dvd drive',
 'DVD Drive',
 5);
INSERT INTO dba.linked_params
("index",
  "display_name",
  "value_data",
  "value_name",
  "link")
VALUES
(12,
 'Item',
 'hard drive',
 'Hard Drive',
 5);
INSERT INTO dba.linked_params
("index",
  "display_name",
  "value_data",
  "value_name",
  "link")
VALUES
(13,
 'Item',
 'zip drive',
 'Zip Drive',
 5);
INSERT INTO dba.linked_params
("index",
  "display_name",
  "value_data",
  "value_name",
  "link")
VALUES
(14,
 'Item',

```



```

        'apple',
        'Apple',
        6);
INSERT INTO dba.linked_params
("index",
 "display_name",
 "value_data",
 "value_name",
 "link")
VALUES
(15,
 'Item',
 'deli',
 'Deli',
 6);
INSERT INTO dba.linked_params
("index",
 "display_name",
 "value_data",
 "value_name",
 "link")
VALUES
(16,
 'Item',
 'ibm',
 'IBM',
 6);
INSERT INTO dba.linked_params
("index",
 "display_name",
 "value_data",
 "value_name",
 "link")
VALUES
(17,
 'Item',
 'memory',
 'Memory',
 7);
INSERT INTO dba.linked_params
("index",
 "display_name",
 "value_data",
 "value_name",
 "link")
VALUES
(18,
 'Item',
 'outlook',
 'MS Outlook',
 8);
INSERT INTO dba.linked_params
("index",
 "display_name",
 "value_data",
 "value_name",

```

```

    "link" )
VALUES
(19,
 'Item',
 'outlook express',
 'MS Outlook Express',
 8);
INSERT INTO dba.linked_params
("index",
 "display_name",
 "value_data",
 "value_name",
 "link")
VALUES
(20,
 'Item',
 'access',
 'Access',
 9);
INSERT INTO dba.linked_params
("index",
 "display_name",
 "value_data",
 "value_name",
 "link")
VALUES
(21,
 'Item',
 'ftp',
 'FTP',
 9);
INSERT INTO dba.linked_params
("index",
 "display_name",
 "value_data",
 "value_name",
 "link")
VALUES
(22,
 'Item',
 'other',
 'Other',
 9);

```

Creating Multilevel Insert Operations for Web Service Mobile Business Objects

Create a multilevel insert operation for two Web service mobile business objects (MBOs).

In this example, you have two MBOs, Order and OrderItem, that both have defined insert operations: the OrderItem.insert operation requires the Order.id, but Order.id is assigned by the enterprise information system (EIS) and not available until the order is created in the EIS. You can create a multilevel insert operation to address this problem. When creating the multilevel insert operation:

- Ensure that Order.insert operation returns a resultSet that has the newly created Order.Id as one of the columns.
 - Chain the two insert operations by creating the appropriate relationship.
 - Ensure the association from Order to OrderItem is from Order.id.
 - Ensure consistent naming: the **Find By** attribute of Order (ID) must match the ID parameter of OrderItem.insert.
1. Create a Web service connection profile to the data source from which you created the MBOs.
 2. Create attributes of the parent MBO (Order). For example, you can drag and drop the Web service data source onto the Mobile Application Diagram, and use the Quick Create wizard to define the MBO.
Define the MBO operation (insert).

Note: Web service multilevel inserts support SOAP bindings only.

3. Click **Finish**.
4. Set or verify the **Fill from attribute** setting:
 - a) In the Mobile Application Diagram, double-click the operation that serves as the insert operation for the parent MBO.
 - b) From the left side of the Properties view, select the **Parameters** tab.
 - c) Verify that each parameter name has a corresponding **Fill from Attribute** value defined.
All parameters of the create operation in the parent MBO and the child MBO must be set to the related **Fill from attribute** value. By default, the related value is set automatically, but in some cases the value cannot be found, so double check the values.
 - d) From the left side of the Properties view, select the **Attributes** tab located on the left, then the **Attributes Mapping** tab located on the top. Locate and select the **Find by** check box for the attribute that serves as the primary-key equivalent for the parent MBO (for example, Id).
5. Create the child MBO (OrderItem) the same way you created the parent – drag and drop the data source onto the Mobile Application Diagram, and follow the Quick Create wizard instructions to create the attributes and operations.
6. From the Properties view, verify that each operation's (insert) parameter name has a corresponding **Fill from attribute** setting.
7. In the Mobile Application Diagram, click **Relationship**, and use the wizard to define a relationship between the MBOs. For example, link the Source object Order "Id" attribute to the Target object OrderItem "Id".
Verify that the child MBO is not syncable, unless you are sure that the child MBO will be synchronized either independently, or through the parent MBO. When the device application designed from these MBOs runs, the child MBOs appear on the Synchronize

screen. If the device-application user attempts to synchronize any of the child MBOs, a `Missing-Sync-Param` exception occurs.

8. Verify that **Filter by** is selected for the child MBO's attribute/parameter used in the relationship.

Understanding Multilevel Insert Operations

In a multilevel insert, multiple mobile business objects are synchronized in a single operation. The mobile business objects must have a defined relationship, and the insert parameters must support the relationship.

Some business processes require multiple related enterprise information system (EIS) operations; for example, creating a sales order with line items. The parent/child relationship is often represented by primary key(PK) / foreign key(FK) attributes in the parent and child mobile business objects (MBOs). When you construct these types of MBOs in an offline client application, the primary-key and foreign-key values are transitory. When EIS operations are called to create real data, the EIS systems generate the actual key values, and the primary key of the parent is copied to the related child MBO creation operations. These types of operations are known as "chained insert" or "multilevel insert."

- For JDBC MBOs using Sybase databases, dragging and dropping a table that contains autoincrement columns (one mechanism for generating primary keys) automatically creates the appropriate operations for obtaining the parent's generated keys and applying them to the children.
- For other EIS types (non-Sybase databases, and applications where key generation does not use the autoincrement technique), you must define the insert operations in such a way that allows the child to obtain the generated keys.

Typically, in a chained-insert operation, you:

1. Create the parent MBO, and indicate the attributes that constitute that MBO's primary key.
2. Create the child MBO and draw a relationship from the parent MBO's primary-key attributes to the child's foreign-key attributes.

Unselect the child MBO's Syncable property, unless you are sure that the child MBO will be synchronized either independently or through the parent MBO. Otherwise, when the device application designed from these MBOs runs, the child MBOs display on the "Synchronize" screen. If the device application user invokes "Synchronize" on any of the child MBOs, a `Missing-Sync-Param` exception occurs.

3. Define the insert operations for the parent and child MBOs.

The insert operation for the parent MBO must return a single row that contains the primary-key values. The column labels must match the attribute names of the parent MBO. With this information, and the relationship-mapping data, Unwired Workspace modifies the input parameters for the insert operation of the child MBOs by replacing the foreign-key attributes with the ones returned from the parent MBO's insert operation. For example:

```
CREATE TABLE parent(pk int autoincrement primary key, p1
varchar(30),...)
CREATE TABLE child(fk int references parent.pk, ...)
```

The parent insert MBO is defined as:

```
INSERT INTO parent(p1, ...) VALUES(?, ...); SELECT * FROM parent
WHERE pk = @@IDENTITY;
```

This batch query inserts the new parent row, and returns a single row containing the newly generated primary-key value.

You must understand the key-generation mechanism used by the EIS application from which you are developing, and be able to determine how to retrieve the newly generated keys during the insert operation (frequently, this logic is wrapped in a stored procedure).

This same technique applies to Web service, SAP, and other EIS systems, though the insert-operation definitions differ.

Note:

- The from attribute of the insert operation parameter is used to infer the foreign-key information of the insert operation parameter. So the name of the attribute (which is the target of the association from the primary key of the parent) and parameter of the insert operation need not be the same.
 - The insert query returns the complete newly generated row, not just the identity column. The single row that is returned must contain all of the columns referenced in the relationship between the parent MBO and the child MBO, and the labels of the columns must match the from attribute names of the parent MBO. Not all columns in the inserted row are required. For example, not all columns are selected or required for a drag-and-drop database operation.
 - A multilevel insert records all logs under the parent MBO. All pending actions are also listed under the parent MBO.
-

Errors may occur if:

- The client sends the parent ID, which does not correspond to the server's interpretation of the parameters of the insert operation.
- The customer's primary key consists of more than one attribute. If the child has multiple foreign-key attributes pointing to the parent, the relationship should list all relevant parent-to-child attributes. As long as the row returned from the parent insert contains all those columns, the child insert should work; all the foreign-key fields are populated from the parent insert result set.
- The insert operation of the parent fails at the back end.
- There is no association relationship between customer and order in which the source attribute/parameter in customer is a primary key and the target parameter in order is a foreign key to customer.

- The result set generated by the parent's insert operation does not have the required single row with the newly created primary key of that operation.

Note: Unwired Server does not report the specific reason of a multilevel insert failure. If you receive errors, or if the insert fails, check each of these items to try and identify the problem.

Modifying SAP Connection Properties

Edit connection properties for an SAP mobile business object, and define the property values as parameters.

You can modify Java connector (JCo) properties in SAP mobile business objects (MBOs), and use the property values as parameters; for example, as personalization keys or default values.

1. In the Mobile Business Object Properties dialog, select the **Connection** tab.
2. Select **Override default authentication**.
3. Under **Connection Properties**, select **Configure as Parameters**.
4. The table displays the configurable connection properties:
 - For each property, you can either enter a value or select a personalization key. You can also create a new personalization key for any property, except the language property, which does not support personalization.
 - The `codepage` property does not appear in the table; it is calculated from the language, and cannot be modified.
 - Set the `Unicode` property to either true or false.
5. Click **Apply**, then click **OK**.

At runtime, you can use SAP connection properties as parameters. For example, if the `SalesOrder.CreateFromData1` operation inserts a sales order for a particular user, and it occurs in the context of a known SAP user, the user's credentials can be passed to the operation.

Documentation Updates

These documentation updates apply to Sybase Unwired Platform version 1.2.1.

Generating Certificates to Enable HTTPS Synchronization

Generate public, private, and identity keys by running the **createcert** command line utility.

Unwired Server and the Afaria server can share a certificate if both products are installed on the same host, or if you create a wildcard certificate (certificate DN is `*.<domain>`). Wildcard certificates may not be accepted by all clients.

1. At a command prompt, change to `<UnwiredPlatform-installDir>\servers\UnwiredServer\SQLAnywhere11\BIN32`.
2. Run:

createcert

- When prompted, enter 1024 as the **RSA key length**. For all remaining prompts, enter appropriate values for your deployment; for example:

```

C:\Sybase\UnwiredPlatform-1_2\Servers\UnwiredServer\SQLAnywhere11\BIN32>createcert
SQL Anywhere X.509 Certificate Generator Version 11.0.0.1578
Enter RSA key length (512-16384): 1024
Generating key pair...
Country Code: US
State/Province: CA
Locality: Dublin
Organization: Sybase
Organizational Unit: ITS
Common Name: SUP
Enter file path of signer's certificate:
Certificate will be a self-signed root
Serial number [generate GUID]:
Generated serial number: cb6a16ef4dd243929446266429cc9107
Certificate valid for how many years (1-100): 2
Certificate Authority (Y/N) [N]:
1. Digital Signature
2. Nonrepudiation
3. Key Encipherment
4. Data Encipherment
5. Key Agreement
6. Certificate Signing
7. CRL Signing
8. Encipher Only
9. Decipher Only
Key Usage [3,4,5]:
Enter file path to save certificate: rsa_public_cert.crt
Enter file path to save private key: rsa_private_cert.crt
Enter password to protect private key: admin123
Enter file path to save identity: rsa_server_identity.crt
C:\Sybase\UnwiredPlatform-1_2\Servers\UnwiredServer\SQLAnywhere11\BIN32>

```

Note: Make a note of your private-key password and identity-key file path; you will need these values again.

Preparing the Unwired Platform Environment for SAP Connections

The SAP JCO connector is used by all Unwired Platform components. Therefore, Sybase recommends that you make all changes concurrently in a distributed environment for development and production installations of Unwired Platform.

Prerequisites

Before you can access the SAP Web site, you must have an SAP account.

These steps describe the common tasks for setting up an SAP environment. Other details, such as where to copy files, differ by component.

- Go to the SAP Web site at <http://service.sap.com/connectors> and download the latest SAP JavaConnector, for example, sapjco-ntintel-2.1.8.
- Unzip the file, which contains:

New Features 1.2.1

- sapjco.jar
 - librfc32.dll
 - sapjcorfc.dll
3. Shut down all Unwired Platform components, including Unwired WorkSpace and all Unwired Servers on your network.
 4. Copy librfc32.dll and sapjcorfc.dll into the following target directories:

Component	Targets
Eclipse Unwired WorkSpace	<ul style="list-style-type: none"> • C:\WINDOWS\system32 • <SUP Installation root>\Unwired-Platform\JDK1.6.0_12\bin
Visual Studio Unwired WorkSpace	<ul style="list-style-type: none"> • \Program Files\Microsoft Visual Studio 9.0\Common7\IDE • <SUP Installation root>\Unwired-Platform\Unwired_WorkSpace\VisualStudio\toolingapi\lib
Unwired Server	<ul style="list-style-type: none"> • <SUP Installation root>\Unwired-Platform\Servers\UnwiredServer\dll

5. Copy sapjco.jar into the following target directories:

Component	Targets
Eclipse Unwired WorkSpace	<ul style="list-style-type: none"> • <SUP Installation root>\Unwired-Platform\Unwired_WorkSpace\Eclipse\sybase_workspace\mobile\eclipse\plugins\com.sybase.uep.com.sap.mw.jco_1.2.0.<version>\lib
Visual Studio Unwired WorkSpace	<ul style="list-style-type: none"> • \UnwiredPlatform\Unwired_WorkSpace\VisualStudio\toolingapi\lib
Unwired Server	<ul style="list-style-type: none"> • <SUP Installation root>\Unwired-Platform\Server\UnwiredServer\lib

6. In a clustered production environment, you must also enable SAP mobile business objects to connect to an SAP R/3 system that uses a router:
 - a) Change to \UnwiredPlatform-1_2\Servers\UnwiredServer\Repository\Instance\com\sybase\sap.
 - b) In a text editor, open <SAPprofile>.properties, where *SAPprofile* is the name of the connection profile that is used to deploy SAP mobile business objects from Unwired WorkSpace to the server.

c) Set the value of the *jco.client.ashost* property to */H/proxyHost/H/applicationServer* , where:

- *proxyHost* is the name of the proxy-server machine, and
- *applicationServer* is the name of the application server

7. After copying the files, restart all components and all Unwired Servers.

Troubleshooting

This troubleshooting information is new for Sybase Unwired Platform version 1.2.1.

Codegen.GetCalledOperations Returns Only One Operation

Problem: In the Client API, **Codegen.GetCalledOperations** returns a list that contains only the first operation.

Solution: Call **FindAll** instead of **GetCalledOperations**.

Device Database Page Size

Problem: For a mobile business object, the sum of the maximum sizes for attributes and parameters cannot exceed 64K; this restriction does not apply to long binary or long varchar datatypes.

If you synchronize metadata from a .NET client to create UltraLite® database tables for a mobile business object with large column sizes, you may see an error similar to `Max row size of UA_10001_10002_param table exceeded. This is analogous to a SQLE_MAX_ROW_SIZE_EXCEEDED (-1132) error.`

On BlackBerry UltraLite (Java) clients, you can synchronize metadata and create database tables, but you may see `end of stream` errors when you synchronize after an insert transaction.

Solution: To work around this problem, try one of these options:

- Increase the page size of the device database, up to 16K. The default size is 4K. You can set the page size using either:
 - An API – see the *Client Object API Cookbook 1.2* on Sybase Unwired Platform Tech Corner, or
 - The Device Application Designer – in the Flow Design Properties view.
- Decrease the size of string datatype columns that are less than 8191 and binary datatype columns that are less than 32767.
- For string columns with a size close to 8191, increase the size to more than 8191, so a long varchar datatype is used instead. For binary columns with a size close to 32767, increase the size to greater than 32767, so a long binary datatype is used instead.

Note: Increasing column sizes may degrade performance.

SAP Connection Error

Problem: When you attempt to connect to an SAP server from the Visual Studio development environment, you may see `Open file C:\WINDOWS\sapmsg.ini failed`, which indicates that the file is missing, and you cannot connect to the SAP system.

Solution: Add `sapmsg.ini` to your Visual Studio environment:

1. In `C:\WINDOWS\`, use a text editor to create a file called `sapmsg.ini` file, and insert this line:

```
<SAP_System_Number>=<IP_Address>
```

Where:

- *SAP_System_Number* is the value of the SAP system number connection property (`jco.client.sysnr`), and
 - *IP_Address* is the IP address of the Unwired Platform machine.
2. Using a text editor, open `C:\WINDOWS\system32\drivers\etc\services`, and add a service that has this format:

```
sapms<SAP_System_Number> <Message_Server_Port>\tcp
```

For example:

```
sapmsC27 3602\tcp
```

3. Restart the system.
4. Verify that group public is enabled for the SAP application. See your SAP documentation.

SAP Parameter-to-Argument Linking

Problem: In an SAP mobile business object, when you use the Detailed Properties dialog to edit operation parameters, arguments and parameter-to-argument links may be lost.

Solution:

1. In the Mobile Application Diagram, right-click **Operation** in the mobile business object, and select **Edit with Wizard**.
2. Restore any missing arguments or links.

Slow Code Generation

Problem: Generating client code for a project with several mobile business objects may take five minutes if you select to automatically generate the user interface.

Solution: On the first page of the Client Code Generation wizard, unselect **Generate GUI frontend from template**.

SQLE_TOO_MANY_PUBLICATIONS Error

Problem: If there are more than 30 syncable mobile business objects (MBOs) in a package, and fewer than 30 syncable MBOs in the Device Application Designer file, a

SQL_E_TOO_MANY_PUBLICATIONS error may occur when you synchronize the MBOs on Windows Mobile.

Solution:

1. In Unwired WorkSpace, open `Program.cs`, which is located in `%Visual Studio solution folder%\%Client project name%`.
2. Set the value of `Sybase.UnwiredPlatform.Data.DatabaseUtilities.UseDynamicPublication` to true.
3. Rebuild and redeploy the project to your Windows Mobile device.

Trailing Space Causes Synchronization Failure

Problem: If you attempt to synchronize a mobile business object (MBO), and its key column includes values that differ only by a space; for example, "test" versus "test ", synchronization fails.

The Enterprise Information System interprets "test" and "test " as different values, and two rows are created. When you synchronize the MBO, the trailing space is stripped from the value "test ", which creates a conflict, and generates an error.

Solution: Do not insert key values that differ only by a space.

Visual Studio 2008 Chinese Edition

Problem: If you are using the Visual Studio 2008 Chinese Edition, you must manually copy some files for the installation to work with Sybase Unwired Platform.

Solution:

1. In the Windows Registry, find your locale ID at `HKEY_CURRENT_USER\Control Panel\International\Locale`.
2. Convert the value of locale ID from hexadecimal to decimal.
3. Copy the files in:
`C:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE\ProjectTemplates\CSharp\Sybase Unwired WorkSpace\1033`
to:
`C:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE\ProjectTemplates\CSharp\Sybase Unwired WorkSpace\localeID`
Where *localeID* is the decimal value of your locale ID.

New Features - Sybase Control Center and Unwired Server

New and enhanced features, updated documentation, and new troubleshooting topics are described.

The tables below provide a brief description of each feature and links to associated topics.

New Sybase Unwired Platform features

Feature	Topics
Install a dedicated consolidated database (CDB, or cache) in an Unwired Server cluster.	<i>Installing a Dedicated Consolidated Database on an Unwired Server Cluster</i> on page 106
Set up client-side certificates to secure synchronization between Microsoft IIS Server and device clients in a Relay Server configuration. Two new synchronization parameters have been added to support this feature: identity and identity_password .	<i>Set Up Client-Side Certificates</i> on page 108
Optimize device application and Unwired Server performance by including a cache update policy when developing mobile business objects.	<i>Cache Update Policy</i> See New Features – Eclipse Edition and New Features – Visual Studio Edition for information about setting cache update policies for MBOs.
Update the consolidated database with changed Enterprise Information System (EIS) using a data change notification (DCN) request. This section provides examples.	<i>Data Change Notification Requests</i> on page 119

Enhanced Sybase Unwired Platform features

Feature	Topics
New Unwired Server configuration properties are available in Sybase Control Center.	<i>Sybase Control Center New Server Configuration Properties</i> on page 122
Updated command line utilities allow you to create and deploy packages from the command line rather than using Sybase Control Center (or Unwired WorkSpace in the development environment).	<i>Command Line Package and Deploy Utilities</i> on page 123
Server-side logging enhancements allow you to configure a separate log4j logger (EISInteractionLogger) to send output to a separate log file (<code>eis.log</code>) or to the consolidated <code>uep.log</code> .	<i>Server Side Logging Enhancement</i> on page 124

Documentation updates

Feature	Topics
Corrected information in steps 1 and 2. In step 1, the directory name is BIN32, not win32. In step 2 , call createcert , not gencert	<i>Generating Certificates to Enable HTTPS Synchronization</i> on page 125
Replace the topic "Configuring Your Environment for SAP" with the topic "Preparing the Unwired Platform Environment for SAP Connections."	<i>Preparing the Unwired Platform Environment for SAP Connections</i>
Deployment Edition postinstallation configuration updates to the topic "Configuring an LDAP Provider for Sybase Control Center" to include information about anonymous log ins and "Setting Up Roles and Passwords in Sybase Control Center" to add role mapping for the Anonymous Login Module to the uafAnonymous role.	<i>Configuring an LDAP Provider for Sybase Control Center</i> on page 127 <i>Setting Up Roles and Passwords in Sybase Control Center</i> on page 128
<p>These Afaria® reference documents, available from the Sybase Unwired Platform topic "Afaria Documentation," have been updated: :</p> <ul style="list-style-type: none"> • Installing Afaria • Afaria Reference Manual Platform • Afaria Reference Manual Components <hr/> <p>Note: Bandwidth throttling notice to all customers</p> <p>Throughout these documents may be textual and graphical references to the bandwidth throttling feature. Although these representations are included in this document, and also appear in the Afaria application, this version of Afaria disables the feature and does not support its use.</p>	On the Sybase Control Center bookshelf, select <i>Administer > Afaria > Afaria Reference > Afaria Documentation</i> .
This topic contains information about how to implement end-to-end security on a device client with HTTPS.	<i>Implementing End-to-End Security for Device Clients</i> on page 129
This topic contains information for how to set the CDB thread count manually.	<i>Setting the CDB Server Thread Count Manually</i> on page 130
Describes counters used in current Unwired Server logging and tracing.	<i>Unwired Server Performance Counter Reference</i> on page 131

Troubleshooting

Feature	Topics
Inability to access the clusterdb results in an incorrect service name for Unwired Server.	<i>Changing Service Names</i> on page 133
Troubleshoot cluster and relay server issues.	<i>Cluster and Relay Server Issues</i> on page 133
Add <code>COMMUNICATION_ERROR_CODE</code> to client applications to debug synchronization errors.	<i>Troubleshooting Communication Errors Between Client Applications and Unwired Server</i> on page 136
Sybase Control Center does not function if it cannot retrieve its list of servers.	<i>Sybase Control Center Issues Retrieving Server List</i> on page 137
Unwired Server reports ODBC errors.	<i>Unwired Server Reports ODBC Connection Problems</i> on page 137
When trying to run a device client with a client-side certificate, errors may be reported, such as -403 HTTP or -413 HTTP.	<i>Resolving Client-Side Certificate Errors</i> on page 138
If you attempt to insert data that exceeds the maximum size defined for the column, an error occurs.	<i>Inserting Data That Exceeds Maximum Size Causes Errors</i> on page 139

New Features

These new features are included in Sybase® Unwired Platform version 1.2.1.

Installing a Dedicated Consolidated Database Within an Unwired Server Cluster

Install a consolidated database (CDB) so that it is independent of any Unwired Server instance in the cluster.

To improve performance, you may want to install a dedicated CDB in a cluster of Unwired Servers so that it is independent of any Unwired Server within that cluster. Typically, as client load increases, the CDB becomes a bottleneck. If a single Unwired Server becomes a bottleneck, you can always add more servers to your cluster, and the load is balanced between all servers. However, you cannot add more CDBs—one CDB handles the entire load for database-related work. A dedicated CDB improves performance because it does not compete with other Unwired Server components for CPU and I/O resources.

See the *Sybase Unwired Platform Installation Guide* for additional information.

1. Install the first Unwired Server in the cluster (using either the Personal Developer Edition or the Enterprise Developer Edition), as a Windows Service. This is the dedicated CDB node.

2. Select the **Create new cluster** option.

This Unwired Server serves as the dedicated CDB for the Unwired Server cluster.

3. After installing the CDB node, disable starting of non-CDB services on the dedicated CDB host:

a) From Windows, select **Start > Settings > Control Panel**.

b) Select **Administrative Tools > Services**.

c) Disable all Unwired Server services except the CDB database (identified as SybaseUnwiredPlaform<hostname>Database1, where *hostname* is the name of the host system), including:

- WatchDog – SybaseUnwiredPlaformhostnameWatchdog1
- Server – SybaseUnwiredPlaformhostnameServer1
- Sybase Unified Agent
- OpenDS LDAP server (for Unwired Server Developer Edition installations)

4. Install other instances of Unwired Server (Deployment Edition) on the other hosts in the cluster.

5. Select **Add to existing cluster** when prompted, and join the cluster to which the dedicated CDB belongs.

Troubleshooting Dedicated Consolidated Database Installation

These instructions are for situations where the procedure for installing a dedicated CDB was not followed in the initial installation resulting in the installation of Sybase Unwired Platform servers with unwanted CDBs attached.

The information about the primary CDB you do not want attached to the SUP installation must be removed from the cluster configuration data (clusterdb) so the SUP service entry can be rebuilt without the CDB dependency.

In this procedure you will remove the CDB in the initial SUP installation (machine A) from the clusterdb, then you will install SUP on a different machine (machine B) and add the new CDB to the existing cluster on machine A.

1. To remove the CDB from the clusterdb configuration on the initial Unwired Server installation (machine A):

a) Verify the SQLAnywhere database is running and that the clusterdb databases are active.

b) Go to the Unwired Server installation directory; for example, C:\Sybase\UnwiredPlatform-1_2\Servers\UnwiredServer and run `updateProps.bat -x`.

The `-x` flag removes this installation from the cluster configuration data.

c) Run:

```
mbservice.bat remove
uadb-service.bat remove
```

2. Install Sybase Unwired Platform on the machine where you want to run the dedicated CDB (machine B). See *Performing a Custom Installation in the Sybase Unwired Platform Installation Guide*.
 - a) In Additional Installation Options, choose **Select to specify cluster configuration options** to add the new CDB to the existing cluster from the initial SUP installation on machine A.
 - b) Choose **Select to run Unwired Server as a service** to start Unwired Server automatically when Windows starts up.

Note: Do not start the SUP servers on machine B at this point.

3. On machine A, shut down the SUP server, including the SQLAnywhere database.
4. Copy *.log and *.db from %SUP_Installation%\UnwiredServer\SQLAnywhere11 on machine A to the same location on the new installation on machine B, overwriting files that have the same name.
5. On machine B, open the Windows Service Manager and set these services:
 - a) Open the Windows Control Panel and click **Administrative Tools > Services**.
 - b) Find SybaseUnwiredPlatform<clustername>Server<number> and set it to Disabled:
 1. Open Services.
 2. Right-click the service that you want to enable or disable, and then click **Properties**.
 3. On the **Log On** tab, click the profile that you want to configure, and click **Disable**.
 4. Click **OK**.
 - c) Find SybaseUnwiredPlatform<clustername>Database<number> and set it to **Autostart**.
6. Start the database service.
7. On machine A, use a text editor to open the sup.properties file, and set sqlany.mode=none. Delete the property sqlany.serverport entirely. This disables the local CDB.
8. Run config.bat.

This adds the initial installation on Machine A back into the cluster. Since the Machine B CDB is already registered, the CDB on Machine A is not added back and its dependencies are removed.

Client-Side Certificates

Set up client-side certificates to secure synchronization between Microsoft IIS server and device clients in a Relay Server configuration. Two synchronization parameters support this feature: **identity** and **identity_password**.

Device clients cannot directly authenticate to Unwired Server using client-side certificates. The client-side certificates can be used only to authenticate to third-party servers and proxies that have been configured to accept client-side certificate authentication.

These synchronization parameters support client-side certificates:

- `identity` – identifies the file that contains the client's identity. An identity consists of the client certificate, the corresponding private key, and, optionally, the certificates of the intermediary certificate authorities. This parameter is equivalent to MobiLink server's 'identity' parameter.
- (Optional) `identity_password` – specifies the password used to encrypt the private key found in the identity file. `identity_passworded` is only required if the private key in the identity file is encrypted. This parameter is equivalent to MobiLink server's 'identity_password' parameter.

Note: Client-side authentication is not supported in UltraLiteJ (RIM Blackberry) devices.

To set up client-side authentication:

1. Set up client authentication for the Virtual Directory/Web Extension that contains the `rs_client.dll`. Request client-side certificates only for connections to the `rs_client.dll`. There should be no request for client-side certificates on the `rs_server.dll` because this is a server connection.
2. Copy the client-side certificate, generated by the Certificate Manager, to the client device.
3. If everything is configured correctly with IIS, then specifying `identity` on the MobiLink client connection parameter should work.

Configure Relay Server with Microsoft IIS using SSL

To use client-side certificates with Sybase Unwired Platform, configure Relay Server with Microsoft Internet Information Services Web server using SSL.

You need to have these installed on your Windows 2003 Server:

- Application server
- Certificate services

You also need to complete these steps from "Configuring the Relay Server with Microsoft Internet Information Services" in the *Configuring the Relay Server With Microsoft IIS using SSL* white paper available on <http://www.sybase.com/detail?id=1059277>:

1. Deploying the Relay Server Web Extensions.
2. Creating an Application Pool.
3. Enabling the Relay Server Web Extensions.

Note: Sybase also recommends that you start the Relay Server as a Windows service.

Generating Client-Side Certificates

Generate a client-side certificate, which is not the same as the one you configure for Unwired Server HTTPS encrypted synchronization.

Prerequisites

These instructions assume you have already configured Unwired Server-side certificates correctly, and that the Unwired Server client will provide a trusted certificate. Windows IIS should be set up with Relay Server for server-side certificates, and should have Application Server and Certificate Services installed as described in Configure Relay Server with Microsoft IIS using SSL available on <http://www.sybase.com/detail?id=1059277>.

Create a new client-side certificate.

1. At a command prompt, change to `<UnwiredPlatform-installDir>\servers\UnwiredServer\SQLAnywhere11\BIN32`.

2. Run:

createcert

3. When prompted, enter 1024 as the **RSA key length**. For all remaining prompts, enter appropriate values for your deployment; for example:

```
C:>createcert
SQL Anywhere X.509 Certificate Generator Version 11.0.1.2250
Enter RSA key length (512-16384): 1024
Generating key pair...
Country Code: CA
State/Province: ON
Locality: Waterloo
Organization: ClientCert
Organizational Unit: ClientCert
Common Name: ClientCert
Enter file path of signer's certificate:
Certificate will be a self-signed root
Serial number [generate GUID]:
Generated serial number: 6d2f67d5a21c4d95a604b701afd37789
Certificate valid for how many years (1-100): 10
Certificate Authority (Y/N) [N]: Y
1. Digital Signature
2. Nonrepudiation
3. Key Encipherment
4. Data Encipherment
5. Key Agreement
6. Certificate Signing
7. CRL Signing
8. Encipher Only
9. Decipher Only
Key Usage [6,7]:
Enter file path to save certificate: rsa_client.crt
Enter file path to save private key: rsa_client.key
Enter password to protect private key: pwd
Enter file path to save identity: id_client.pem
```

Note: You must use an RSA Transport Layer Security (TLS) certificate, and not a certificate generated from ECC TLS or from another source such as **openssl** or **createcert** in an SQL Anywhere installation.

Note: Make a note of your private-key file path and password values (`rsa_client.key` and `pwd`), and the certificate and identity file paths (`rsa_client.crt` and `id_client.pem`). You will need these values again.

Installing the CA Certificate

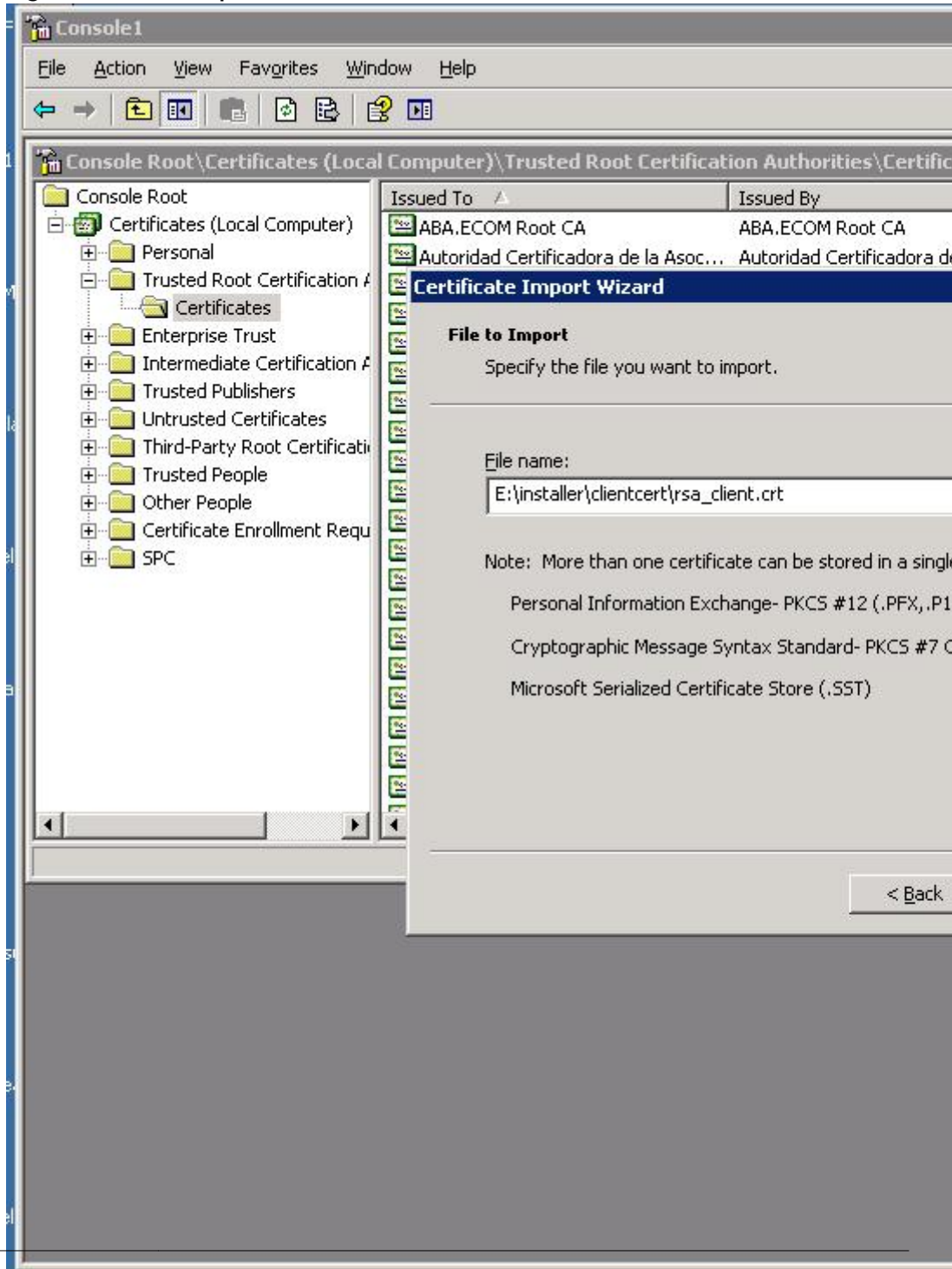
Install the root CA certificate (`rsa_client.crt`) on the Web server machine to enable the Web server to trust the Web site certificate installed on the IIS Web site.

Install the root CA certificate in the Trusted Root Certification Authorities store on the Web server machine (this is the `rsa_client.crt` file that you created in the Generating Client-Side Certificate step). This enables the Web server to trust the Web site certificate installed on the IIS Web site.

Install the root CA certificate into the Web server certificate store:

1. Click **Start > Run**, and enter **mmc**. Click **OK**.
2. In the Console1 window, select **File > Add/Remove Snap-in**.
3. In the Add/Remove Snap-in dialog, click **Add**.
4. In the Add Standalone Snap-in dialog, select **Certificates**, and click **Add**.
5. On the Certificates snap-in page, select **Computer account**, and click **Next**.
6. On the Select Computer page, select **Local computer**, and click **Finish**.
7. Click **Close**, and then **OK**.
8. Import the certificates:
 - a) Expand the Certificates node, and then expand the Trusted Root Certification Authorities node.
 - b) Right click **Certificates**, select **All Tasks**, and click **Import**.
9. Click **Next** on the Welcome to the Certificate Import Wizard page.
10. On the File to Import page, click **Browse**, and specify the `rsa_client.crt` file path. This is the `rsa_client.crt` file that you created in the Generating Client-Side Certificate step.

Figure 5: Client Properties - Secure Communications



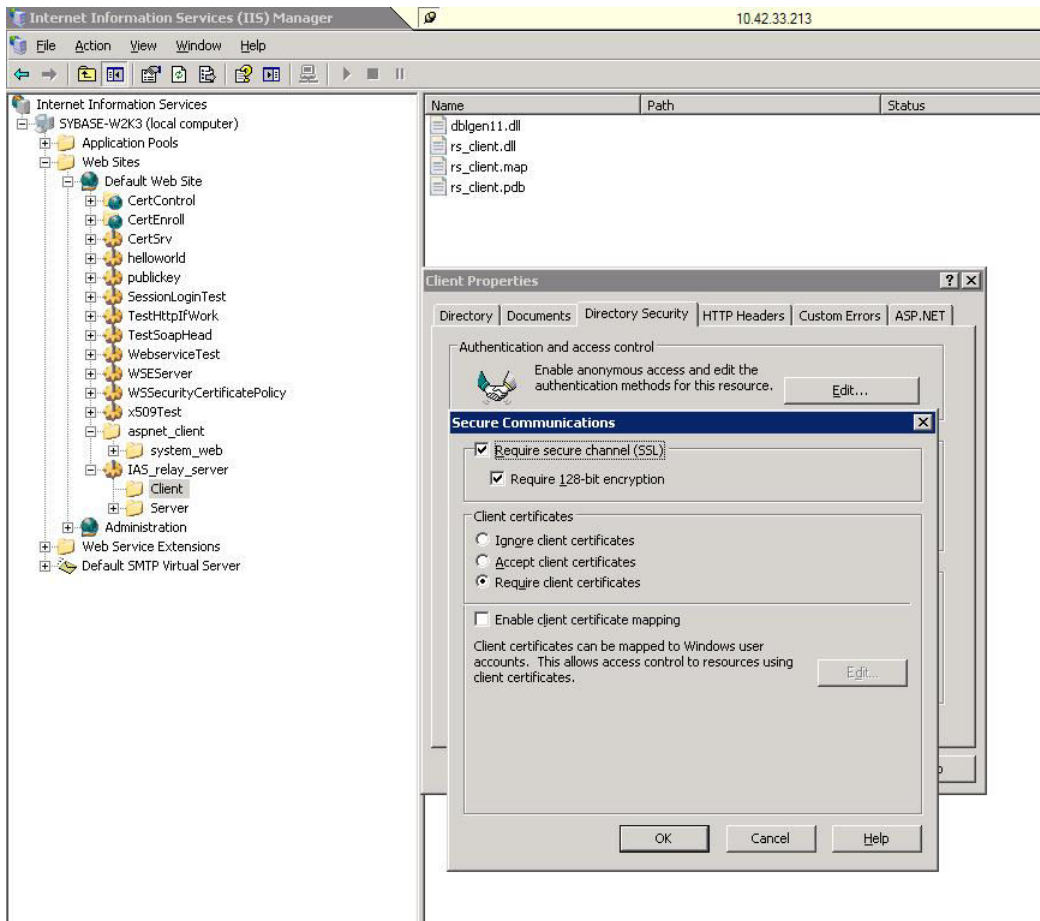
Click **Next**.

11. On the Certificate Store page, accept the default setting, Place All Certificates In The Following Store, and click **Next**.
12. Click **Finish** on the Completing the Certificate Import page.
13. Click **OK** in the Certificate Import Wizard dialog box informing you that the import was successful.

Setting Up RS Client with Client-Side Certificates

Set up rs_client properties in Microsoft Internet Information Services (IIS). These properties are required for client-side certificate authentication.

1. Click **Start > Settings > Control Panel > Administrative Tools**, and select Internet Information Services (IIS) Manager.
2. Right click the client folder under ias_relay_server, and select **Properties**.
3. Select the **Directory Security** tab.
4. Select **Edit**.
5. Select:
 - Require Secure Channel (SSL)
 - Require 128-bit Encryption
 - Require Client Certificates



Click **OK**.

6. Click **Apply**, and then click **OK**.

7. To use client-side certificate authentication:

- a) Go to **Client Properties > Directory Security > Authentication and Access Control > Edit**.
- b) Unselect Integrated Windows Authentication.

Running the Client

To run the client using the certificate, copy the certificate files to your client machine, laptop, or device, and add client-side certificate related information to the user profile.

Note: This procedure is typical in a production environment. In a development or test environment, you could also configure the profile from Unwired WorkSpace. See [Configuring a Profile from a Generated Project](#).

1. Copy the certificate files `rsa_root.crt` and `id_client.pem` to your client machine, laptop, or device. These files were created in the Generating Client-side Certificates step. This example shows the Win32 client.

Note: Client-side certificates are not supported in UltraLiteJ (BlackBerry devices).

2. On the client machine, laptop, or device, access the New Profile dialog.
3. Include the client-side certificate information in the profile, including:
 - MobiLink Port – this example shows 443.
 - ML Protocol – this example uses HTTPS (this is equivalent to the MobiLink Stream Type).
 - Stream Parameter – this example uses the .NET Win32TestClient:


```
"url_suffix=/ias_relay_server/client/rs_client.dll/
[SUP_FARM_ID];tls_type=RSA;trusted_certificates=rsa_root
.crt;identity=id_client.pem;identity_password=pwd;"
```

For this example, you would also copy the `rsa_root.crt` and `id_client.pem` files into `... \Win32TestClient \bin \Debug`.

The screenshot shows a 'New Profile' dialog box with the following fields and values:

Field	Value
Name	myprofile
Username	supAdmin
Password	XXXXXXXXXX
Mobilink Host	10.44.148.110
MobiLink Port	443
Package	Mobile Studio
ML Protocol	HTTPS
Stream Parameter	t.pem;identity_password=pwd

Buttons: Save, Cancel

4. If you encounter an error, take action based on the error reported. See Resolving Client-Side Certificate Errors.

Configuring a Profile from a Generated Project

In a development or test environment, you can set up the client-side certificate by configuring the profile from the generated Windows Mobile project in Visual Studio Edition, instead of from the client machine, laptop, or device.

You can also copy the certificates to the client, and set up or modify the profile to use the certificates, from Unwired WorkSpace instead of from the client machine. Keep in mind the following guidelines:

- HTTPS is only established between HTTPS endpoints. The example values entered in WorkSpace for a generated project would only be used in the development or test environment. These values would not automatically deploy into a production environment.
- A Relay Server is used as an HTTPS endpoint in the example below, and would make a convenient endpoint for a development or test environment. But your production environment may vary (for example, you may use a third-party HTTPS solution that includes hardware load balancers, reverse proxies, and others).

Note: Client-side certificates are not supported in UltraLiteJ (BlackBerry devices).

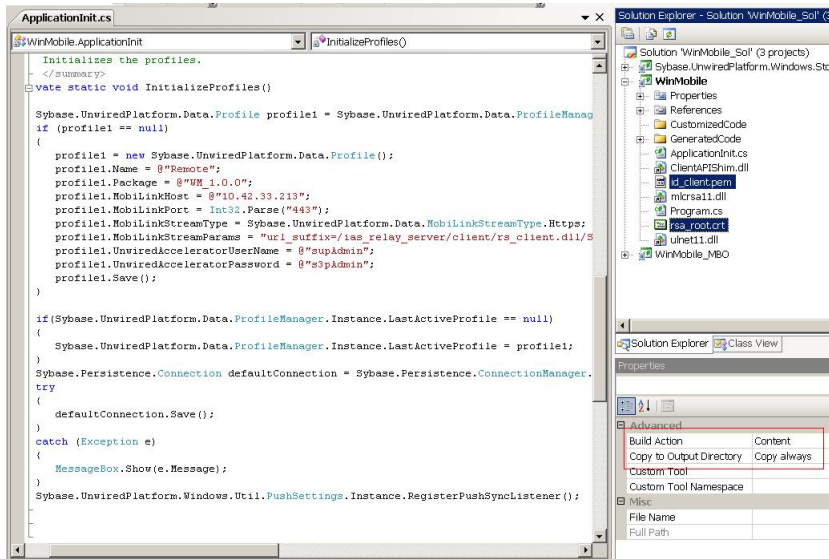
1. Change the connection profile:

1. Open the generated Windows Mobile project.
2. Open the `ApplicationInit.cs` file.
3. Locate the **InitializeProfiles** method.

2. Configure the profile:

```
profile1.MobiLinkPort = Int32.Parse("443");
profile1.MobiLinkStreamType =
Sybase.UnwiredPlatform.Data.MobiLinkStreamType.Https;
profile1.MobiLinkStreamParams = "url_suffix=/ias_relay_server/
client/rs_client.dll/[SUP_FARM_ID];
tls_type=RSA;trusted_certificates=[Output_File_Folder]\
\rsa_root.crt;identity=[Output_File_Folder]\\id_client.pem;
identity_password=pwd;";
```

Note: The `Output_File_Folder` is where the project is deployed to an emulator or device. If the project output file folder is `%CSIDL_PROGRAM_FILES%\WinMobile_Sol`, then set `trusted_certificates` equals `"\\Program Files\\WinMobile_Sol\rsa_root.crt"`.



3. Add the certificate files:

1. Right-click the project, and select **Add > Existing Item**.
2. Find the two files `rsa_root.crt` and `id_client.pem`, and add them to the project.
3. In Solution Explorer, choose the two files, open the Properties window, and change Build Action to "Content" and Copy to Output Directory to "Copy always."
4. If you encounter an error, take action based on the error reported. See Resolving Client-Side Certificate Errors.

Cache Update Policy

Fine-tune device application and Unwired Server performance by defining a cache update policy for mobile business object operations.

Setting a cache update policy for mobile business object (MBO) operations gives you more control of both Unwired Server interactions with the enterprise information system (EIS) to which the MBO is bound, and consolidated database updates. Fine-tuning these interactions and updates improves both Unwired Server and device application performance.

Note: Consolidated database, CDB, and cache all refer to the same thing, and the terms are used interchangeably.

- MBO operations perform specific functions based on their definition:
 - Primary read operation – the EIS operation used to define and initially populate the CDB (from the EIS) for the MBO.
 - Create, update, delete (CRUD operations) – modify EIS data depending on the definition of the operation. Unwired Server maintains a cache (CDB) of back-end EIS data to provide differential synchronization and to minimize EIS interaction. When an

operation is submitted from a device application to the EIS, the cache must be refreshed.

While these types of bulk-fetch and CDB caching are effective in reducing the number of interactions required with the back-end EIS, and work well in some other cases (where MBO data is occasionally updated in the back-end), performance suffers if changes are initiated from Unwired Server (by way of MBO operations), or if changes are frequent.

The cache update policy introduces alternative methods of updating the cache at finer granularity, which improves performance.

- Alternate read operations – can be invoked either from:
 - A chained read cache policy to augment CUD operations by chaining an alternate read operation to a CUD operation.
 - A data change notification, which provides a mechanism to invoke MBO operations, including alternate read operations. This mechanism is independent of a cache update policy.
- Cache update policy – determines how the CDB is updated after an operation. You can set the cache update policy for operations, with these exceptions:
 - Operations defined as "Other" do not support alternate read or a cache update policy.
 - When invoked, alternate read operations always use the apply operation results cache update policy.

Versions of Sybase Unwired Platform earlier than 1.2.1 supported only the invalidate cache policy—any CUD or other operation issued from a device application invalidated the cache and required a primary read operation to refresh the cache.

In Unwired Platform version 1.2.1, these are the five cache update policies you can associate with MBO CUD operations:

- Invalidate cache
- No invalidate cache
- Apply operation result
- Apply operation parameters
- Chained read

When an MBO CUD operation is called, its cache update policy determines how operation results are applied to the consolidated database. Generally, there are two ways of calling an MBO operation:

1. Device client calls the operation.
2. A data change notification (DCN) request contains the operation.

Note: Other methods used to update the CDBs that are external to MBO operations, and not associated with cache update policies include:

1. EIS-initiated DCN – an HTTP request to Unwired Server, in which the DCN request contains the payload (information about the changed data).

Note: EIS-initiated DCN also supports HTTP POST requests which provides a higher level of security.

2. Scheduled data refresh – defined in Sybase Control Center; polls the EIS for changes at specified intervals.
-

Data Change Notification Requests

The consolidated database (or cache) can be updated by changed data at the enterprise information system (EIS) level using a data change notification (DCN) request. DCN requests can invoke arbitrary MBO operations, including alternate read operations. DCN requests are typically initiated by the EIS back end.

A DCN request is primarily meant to supply the changed data (at the back end) to Unwired Server as payload to enable updating of Unwired Server cache. The payload can contain inserts, updates, and deletions from the back end.

Unwired Server expects a DCN request via HTTP or HTTPS. The base URL for the request is:

```
http://<host>:<port>/onepage/servlet/UWPServlet
```

The mandatory parameters for this URL are:

- app=uep
- cmd=ServerAdmin.dcn
- authenticate.user=<username>
- authenticate.password=<password>
- dcn_request=<dcn request>

Though the authenticate.user, authenticate.password and dcn_request parameters can be provided as URL parameters, Sybase recommends that, for security purposes, you instead embed them in the POST body.

DCN requests must be in a specified in a JavaScript Object Notation (JSON) format, which must include:

- Package name (pkg)
- A list of messages (messages), where each message includes:
 - A message ID (id) used to report back the status.
 - Mobile business object name (mbo).
 - Operation name (op): ":upsert", ":delete", or an operation name of the specified MBO. ":upsert" and ":delete" are special operation names to specify payload data.
 - Bindings (cols): a list of name/value pairs covering at least all primary key attributes of the MBO. For payload data, the bindings contain the payload specified as MBO attribute/parameter names and their values. For an MBO operation, bindings contain operation parameter names and values.

New Features 1.2.1

The format of non-string data is same as the one used in providing default values of parameters in tooling. For example, specify timestamp values in a format similar to 2009-03-04T17:03:00+05:30. Base binding names on MBO names rather than EIS names.

For MBO operations that are executed using DCN, any explicit value specified in the DCN request takes precedence over a personalized value. To use a personalized value in DCN processing, do not include the personalized parameter name and its value in the DCN request. The personalized value is then inferred by the server and added to the supplied bindings. Similarly, omit parameters with default values if the intent is to use previously specified default values.

This sample DCN includes new lines and indentation only for clarity; you need not include them in an actual request:

```
dcn_request=  
  {"pkg": "Mobile Studio",  
   "messages": [  
     {  
       "id": "1",  
       "mbo": "company",  
       "op": ":upsert",  
       "cols":  
         {  
           "company_id": "6",  
           "name": "Sonia",  
           "so_user": "cjobson"  
         }  
     },  
     {  
       "id": "2",  
       "mbo": "contact",  
       "op": ":upsert",  
       "cols":  
         {  
           "company_id": "7",  
           "first_name": "Diana",  
           "contact_id": "5"  
         }  
     }  
   ]  
}
```

For providing null values to parameters, the special value null should be user, note the absence of quotes -

```
"cols":  
{  
  "company_id": "6",  
  "name": null,  
  "so_user": "cjobson"  
}
```

The old value parameters can be provided in the DCN request by prepending the parameter name with "old:" as follows (marked in italics) -

```
"cols":
{
  "company_id": "6",
  "name": "Changed",
  "old:name": "Diana",
  "so_user": "cjobson"
}
```

Data Change Notification Results

Each binding in a data change notification (DCN) request is associated with an ID. The result status of the DCN request is returned in JavaScript Object Notation (JSON) format. Basically it is a list of IDs followed by a Boolean success field and status message, in case of error.

This example shows The exact JSON format for a DCN result for a request of three IDs (recID1, recID2, recID3). The example has been formatted using new lines, and indentations, which are not present in an actual response:

```
[
  {
    "recordIDs":
    [
      "recID1",
      "recID2"
    ],
    "success": true,
    "statusMessage": ""
  },
  {
    "recordIDs":
    [
      "recID3"
    ],
    "success": false,
    "statusMessage": "bad msg2"
  }
]
```

Data Change Notification Filters

Data change notification (DCN) requests need not always be in the format Unwired Server expects. You can deploy a DCN filter to Unwired Server and reference it in the DCN request. Unwired Server allows the filter to preprocess the submitted DCN. The filter converts raw data in the DCN request to the required JavaScript Object Notation (JSON) format. The filter can also postprocess the JSON response returned by the Unwired Server into the format preferred by the back end (which is governed by the implementation in the filter class).

The filter interface `DCNFilter` is located in the `com.sybase.sup.dcn` package in the `onepage.jar` file. All classes that implement a DCN filter should implement this interface. The functions available in the interface are:

- **String preprocess(String blobDCNRequest, Map<String, String> requestHeaders);**
– takes the DCN request as a binary large object (blob), converts it into a valid JSON DCN request format and returns the same.
- **String postprocess(String jsonDCNResult, Map<String, String> responseHeaders);**
– takes the DCN result in a valid JSON format, converts it to the EIS-specific format and returns the same.

When specifying filters, a `dcn_filter` parameter to the base URL, and to the parameters specified in the DCN request section. The `dcn_filter` parameter specifies the fully classified name of the filter class, which must be in a valid CLASSPATH location so Unwired Server can locate it using its fully qualified name. If the filter class is in Unwired Platform's root folder, it is automatically propagated across a cluster. The complete URL for specifying a DCN request with a filter is:

```
http://<host>:<port>/onpage/servlet/UWPServlet?
app=uwp&cmd=ServerAdmin.dcn&authenticate.user=<username>
&authenticate.password=<password>&dcn_filter=
<fully qualified name of the filter class>&dcn_request=
<dcn request>
```

The `dcn_request` is in a format that is specific to the back end. The filter class can preprocess to the JSON format expected by Unwired Server.

Data Change Notification, Push Notification, and Filter by Partitioning Examples

User scenarios illustrate how Data Change Notification (DCN) and Unwired Server Push Notifications work together to playback and filter data.

Enhanced Features

These feature are enhanced in Sybase Unwired Platform version 1.2.1.

Sybase Control Center New Server Configuration Properties

Beginning in version 1.2.1, you can use Sybase Control Center to configure properties for MobiLink server and SQL Anywhere.

These two properties are in the Server Configuration section in Sybase Control Center. Click the relevant tab to see the option.

Optional property	Server Configuration tab	Description
sqlany.useroptions	CDB	Allows free-form specification of additional command line options for SQL Anywhere server.
sup.sync.useroptions	Synchronization	Allows free-form specification of additional command line options for the MobiLink server.

Command Line Package and Deploy Utilities

Create and deploy packages from the command line rather than using either Unwired WorkSpace or Sybase Control Center.

You need JDK1.6.0_10 or higher to use these utilities. Also ensure that you have set the JAVA_HOME environment variable to the installation location of this JDK version. Otherwise, you must use a text editor to modify the existing JDK path in the batch file itself.

- Use the `package.bat` utility to create a package on the Unwired Server or Unwired WorkSpace host.
- Use the `deploy.bat` utility to deploy a deployment unit to Unwired Server.

`deploy.bat` and `package.bat` have been changed for SUP. These changes include:

- The `readme.txt` file has been replaced with `DeployReadMe.txt`, which describes the utilities.
- The utilities are available in `uep-server.zip`, `UEPAdminClient.zip` and `uep12ebf1.zip` files.
- A new class, `com.sybase.uep.admin.common.CommandLineDeployment` has been added.
- `deploy.bat` automatically creates the specified package name in the Deployment Unit (DU) if it does not exist.
- The default deployment mode is REPLACE.

Packaging and Deploying MBOs from the Unwired Server Host

Use command line utilities to package and deploy mobile business objects (MBOs) as an alternative to Sybase Control Center (SCC).

1. Open a command prompt.
2. Change to the `<unwired-platform>\bin` folder.
3. Run the batch file.
4. Supply utility property values as required.

Packaging and Deploying MBOs from Unwired WorkSpace

Use command line utilities as an alternative to using Unwired WorkSpace to package and deploy mobile business objects (MBOs).

1. From Unwired WorkSpace, create a mobile deployment package that contains the MBOs you want to deploy.

There are several methods you can use (for example, select **File > New > Mobile Deployment Package**), as long as the result is a deployment package (a file with a `.suppkgdef` extension) in the mobile application's project Deployment subfolder.

2. From WorkSpace Navigator, expand the **Project** folder and **Deployment** folder, right-click `.suppkgdef` and select **Build Package (Full)**.

The build process generates a file named `<deployment_name>.suppkg`, where `<deployment_name>` is the name of the deployment package, that contains two files:

- `deployment_descriptor.xml`
- `deployment_unit.xml`

3. Unzip the `<deployment_name>.suppkg` file.

For command line packaging and deployment, you need only the `deployment_unit.xml` file.

4. Modify the `deployment_unit.xml` file as needed.

For example, to change the package name to provide version information, change:
`package_name="CustomerTest_1.0.0"`

5. To create the package from the command line, go to the directory where `package.bat` resides and run:

```
package <packagename> <host> <port> <loginid> <password>
```

6. To deploy the package name that contains the `deployment_unit.xml` file, go to the directory where `deploy.bat` resides and run:

```
deploy <deploymentUnitFileName> <host> <port> <loginid>  
<password> <deployMode>
```

Server-Side Logging

Server-side logging allows configuration of a separate log4j logger (EISInteractionLogger) to send output to a separate log file (`eis.log`) or to the consolidated `uep.log`.

This feature allows administrators and developers to correlate server side log4j logs to the user request.

The configuration of the logger is managed by directly editing the `log4j.properties` file. By default, `log4j.properties` logs in the `eis.log` at the WARN level.

A synchronization request can have zero or more enterprise information system (EIS) interactions depending on playback policy, uncommitted transactions, and mobile business object (MBO) associations spidering. One logging statement is written for each EIS interaction (both MBO and operations) using the new logger. If the EIS request is successful, logging is done in INFO mode, for example:

```
INFO: SUCCESS:<Userid>:<MBO name>[.Operation name]:<effective  
parameter-value pairs>: [Warning message from EIS]
```

When an EIS request fails, a WARNING is logged, for example:

```
WARN: FAILED:<Userid>:<MBO name>[.Operation name]:<effective  
parameter-value pairs>: <Error message from EIS>
```

An error message from the EIS means that the request failed, while the warning means that the EIS request succeeded but with warnings. A failure log is always accompanied by an error message from the EIS, while a success log may optionally have a warning message. Currently, only the warnings from the SAP EIS are processed.

Documentation Updates

These documentation updates apply to Sybase Unwired Platform version 1.2.1.

Generating Certificates to Enable HTTPS Synchronization

Generate public, private, and identity keys by running the **createcert** command line utility.

Unwired Server and the Afaria server can share a certificate if both products are installed on the same host, or if you create a wildcard certificate (certificate DN is *.<domain>). Wildcard certificates may not be accepted by all clients.

1. At a command prompt, change to `<UnwiredPlatform-installDir>\servers\UnwiredServer\SQLAnywhere11\BIN32`.
2. Run:
createcert
3. When prompted, enter 1024 as the **RSA key length**. For all remaining prompts, enter appropriate values for your deployment; for example:

```

C:\Sybase\UnwiredPlatform-1_2\Servers\UnwiredServer\SQLAnywhere11\BIN32>createcert
SQL Anywhere X.509 Certificate Generator Version 11.0.0.1578
Enter RSA key length (512-16384): 1024
Generating key pair...
Country Code: US
State/Province: CA
Locality: Dublin
Organization: Sybase
Organizational Unit: ITS
Common Name: SUP
Enter file path of signer's certificate:
Certificate will be a self-signed root
Serial number [generate GUID]:
Generated serial number: cb6a16ef4dd243929446266429cc9107
Certificate valid for how many years (1-100): 2
Certificate Authority (Y/N) [N]:
1. Digital Signature
2. Nonrepudiation
3. Key Encipherment
4. Data Encipherment
5. Key Agreement
6. Certificate Signing
7. CRL Signing
8. Encipher Only
9. Decipher Only
Key Usage [3,4,5]:
Enter file path to save certificate: rsa_public_cert.crt
Enter file path to save private key: rsa_private_cert.crt
Enter password to protect private key: admin123
Enter file path to save identity: rsa_server_identity.crt

C:\Sybase\UnwiredPlatform-1_2\Servers\UnwiredServer\SQLAnywhere11\BIN32>

```

Note: Make a note of your private-key password and identity-key file path; you will need these values again.

Preparing the Unwired Platform Environment for SAP Connections

The SAP JCO connector is used by all Unwired Platform components. Therefore, Sybase recommends that you make all changes concurrently in a distributed environment for development and production installations of Unwired Platform.

Prerequisites

Before you can access the SAP Web site, you must have an SAP account.

These steps describe the common tasks for setting up an SAP environment. Other details, such as where to copy files, differ by component.

1. Go to the SAP Web site at <http://service.sap.com/connectors> and download the latest SAP JavaConnector, for example, `sapjco-ntintel-2.1.8`.
2. Unzip the file, which contains:
 - `sapjco.jar`
 - `librfc32.dll`
 - `sapjcorfc.dll`
3. Shut down all Unwired Platform components, including Unwired WorkSpace and all Unwired Servers on your network.
4. Copy `librfc32.dll` and `sapjcorfc.dll` into the following target directories:

Component	Targets
Eclipse Unwired WorkSpace	<ul style="list-style-type: none"> • <code>C:\WINDOWS\system32</code> • <code><SUP Installation root>\Unwired-Platform\JDK1.6.0_12\bin</code>
Visual Studio Unwired WorkSpace	<ul style="list-style-type: none"> • <code>\Program Files\Microsoft Visual Studio 9.0\Common7\IDE</code> • <code><SUP Installation root>\Unwired-Platform\Unwired_WorkSpace\VisualStudio\toolingapi\lib</code>
Unwired Server	<ul style="list-style-type: none"> • <code><SUP Installation root>\Unwired-Platform\Servers\UnwiredServer\dll</code>

5. Copy `sapjco.jar` into the following target directories:

Component	Targets
Eclipse Unwired WorkSpace	<ul style="list-style-type: none"> <SUP Installation root>\Unwired-Platform\Unwired_WorkSpace\Eclipse\sybase_workspace\mobile\eclipse\plugins\com.sybase.uep.com.sap.mw.jco_1.2.0.<version>\lib
Visual Studio Unwired WorkSpace	<ul style="list-style-type: none"> \UnwiredPlatform\Unwired_WorkSpace\VisualStudio\toolingapi\lib
Unwired Server	<ul style="list-style-type: none"> <SUP Installation root>\Unwired-Platform\Server\UnwiredServer\lib

6. In a clustered production environment, you must also enable SAP mobile business objects to connect to an SAP R/3 system that uses a router:
 - a) Change to \UnwiredPlatform-1_2\Servers\UnwiredServer\Repository\Instance\com\sybase\sap.
 - b) In a text editor, open <SAPprofile>.properties, where *SAPprofile* is the name of the connection profile that is used to deploy SAP mobile business objects from Unwired WorkSpace to the server.
 - c) Set the value of the *jco.client.ashost* property to /H/*proxyHost*/H/*applicationServer* , where:
 - *proxyHost* is the name of the proxy-server machine, and
 - *applicationServer* is the name of the application server
7. After copying the files, restart all components and all Unwired Servers.

Configuring an LDAP Provider for Sybase Control Center

Configure an LDAP Provider for Sybase Control Center (Deployment Edition) by editing the security properties file to point to the correct LDAP server.

1. Open the <UAF-install-dir>\conf\csi.properties file.
2. Add the production server to this section. Replace <com.mySybProductPkg> with the package prefix that is appropriate for you Sybase product.

For example:

```
CSI.loginModule.
7.options.AuthenticationSearchBase=UO=users,dc=example,dc=com

CSI.loginModule.7.options.BindDN=cn=Directory Manager
CSI.loginModule.7.options.BindPassword=secret
CSI.loginModule.7.options.DefaultSearchBase=dc=example,dc=com

CSI.loginModule.7.options.ProviderURL=ldap://localhost:10389
CSI.loginModule.
7.options.RoleSearchBase=ou=groups,dc=example,dc=com
```

```
CSI.loginModule.7.options.ServerType=openldap
CSI.loginModule.7.options.moduleName=LDAP Login Module
CSI.loginModule.7.provider=<com.mySybProductPkg>.LDAPLoginModule
```

3. To allow anonymous logins, include the Anonymous Login Module in the `<UAF-install-dir>\conf\csi.properties` file:

```
# Anonymous Login Module
CSI.loginModule.
0.provider=com.sybase.ua.services.security.anonymous.AnonymousLoginModule
CSI.loginModule.0.controlFlag=sufficient
CSI.loginModule.0.options.moduleName=Anonymous Login Module
CSI.loginModule.0.options.roles=uaAnonymous
```

4. Save the file.

Next

Configure the Sybase Unwired Platform users and passwords for Sybase Control Center.

Setting Up Roles and Passwords in Sybase Control Center

Set the initial user roles and passwords required for Sybase Control Center administrator login.

1. Comment out this line if it exists:

```
CSI.loginModule.7.options.roles=SUP Administrator
```

Otherwise, anyone that authenticates with a valid user name and password (including those mapped to supUser) obtains the SUP Administrator role.

2. Ensure that the roles defined in the LDAP repository match the roles defined in the `<UAF_InstallDir>\conf\roles-map.xml` file.

By default, the role mapping file contains the following roles in the LDAP Login Module definition:

```
<role-mapping modRole="SUP Administrator"
uafRole="uaAgentAdmin" />
<role-mapping modRole="SUP Developer" uafRole="uaGuest" />
<role-mapping modRole="SUP User" uafRole="uaGuest" />
```

3. To add role mapping for the Anonymous Login Module to the uaAnonymous role:
 - a) Add the uaAnonymous role to the `<uaf-roles>` section of the `roles-map.xml` file:

```
<role name="uaAnonymous" description="Anonymous role" />
```

- b) Add the role mapping in the `<security-modules>` section of the `roles-map.xml` file:

```
<module name='Anonymous Login Module'>
    <role-mapping modRole='uaAnonymous'
uafRole='uaAnonymous' />
</module>
```

4. Set the BindPassword and ProviderURL properties with values used in your deployment.
5. To encrypt sensitive values inside this file, use the CSI tool utility. For example, you could use the following command to encode the password "ldapPwd".

```
java -jar <UnwiredPlatform_InstallDir>\Servers\
UnwiredServer\SQLAnywhere10\java\csi-tool.jar
csi.encmessage @tomcat\conf\csibootstrap.properties
--text %ldapPwd%
```

The encrypted password displays on the screen.

6. Cut and paste this into the CSI configuration file.

For example, if you supply a real password as represented by "ldapPwd" in the example for step 4, the tool might display the following result:

```
CSITool 3.1 ©) Copyright 2005-2007 Sybase, Inc. (3.1M5/2008/07/29
14:40:08PDT)
Running task: csi.encmessage
Successfully encrypted message. The output was sent to stdout.
1-AAAAGgQQOLL
+LpXJO8fO9T4SrQYRC9lRT1w5ePfdczQTDsP8iACk9mDAbm3F3p5a1wXWKK8+NdJukn
c7w2nw5aGJlyG3xQ==
```

The string you copy is bolded in the output above.

Implementing End-to-End Security for Device Clients

Use the **createkey** utility to create a key pair, then edit the `configure-sup.xml` file to implement end-to-end encryption on device clients with HTTPS.

1. Create a key pair for use end-to-end encryption by running `createkey.exe`, located in `Sybase\UnwiredPlatform-1_2\Servers\UnwiredServer\SQLAnywhere11\BIN32`.

For more information go to Sybase Product Manuals () SQL Anywhere 11.0.1 > SQL Anywhere Server - Database Administration > Administering Your Database > Database Administration Utilities > Key Pair Generator Utility (createkey)

2. Use a text editor to modify `configure-sup.xml`, located in `Sybase\UnwiredPlatform1_2\Servers\UnwiredServer:`

```
<target name="mlserver.xoptions.https" if="is.https.protocol"
depends="import_encrypted_supprops">
    <property name="mlserver.xoptions" value="$
{sup.sync.protocol}{port=${sup.sync.httpsport};identity=${
{sup.sync.certificate};
identity_password=${
{decrypted.sup.sync.certificate_password};e2ee_type=rsa;e2ee_priv
ate_key=<location_of_generated_private_key>;
e2ee_private_key_password=sybase)" />
```

3. Use the ConnectionManager API to add the trusted certificate for the device client, for example:

```
Connection c = new Connection();
c.MobiLinkHost = "sup.sybase.com";
c.MobiLinkPort = 2440;
c.MobiLinkStreamType = MobiLinkStreamType.Https;
c.MobiLinkStreamParams="trusted_certificates=full_path_to_Public
cert.crt;e2ee_type=rsa;e2ee_public_key=
<location of generated public key>";
c.Name = "supAdmin";
c.Package = "Customer_1.0.0";
c.UserName = "supAdmin";
c.Password = "supAdmin";
c.Save()
```

See *Sybase Unwired Platform 1.2 > Sybase Control Center 1.2 > Administer > Unwired Server > Manage > Server Configuration > Configuring Unwired Server Properties > Synchronization > Generating Certificates for HTTPS-Enabled Synchronization > Connection to Unwired Server Using Windows-based Mobile Device Clients.*

4. Synchronize the MBOs.

Setting the CDB Server Thread Count Manually

If the CDB is not installed on the same machine as Unwired Server, you must set the thread count for the CDB manually. There may also be other situations when you want to set the thread count manually, outside of Sybase Control Center.

Setting the CDB thread count in the sup.properties file

1. Use a text editor to open the `sup.properties` file, located in Sybase \UnwiredPlatform1_2\Servers\UnwiredServer, and set the thread count in the `sqlany.threadcount` property.

Note: This value must be 5 units higher than `ml.threadcount`.

2. Save and close `sup.properties`.
3. To register the changes in the clusterdb, change to Sybase \UnwiredPlatform1_2\Servers\UnwiredServer, and run:

```
updateProps.bat -r
```
4. From the Windows Start menu, select **Settings > Control Panel > Administrative Tools > Services**, and shut down the Sybase services.
5. If the CDB is running as a service, reinstall the service by running:

```
uadb-service.bat install
```
6. Restart Unwired Platform services:
 - a. From the Windows Start menu, select **Settings > Control Panel > Administrative Tools > Services**.
 - b. For each service, click **Start**.

Note: Restart Unwired Platform services in this order:

1. SybaseUnwiredPlatformDatabase<*install_number*> (Main (consolidated) database for Sybase Unwired Platform runtime)
 2. SybaseUnwiredPlatformServer<*install_number*>
 3. SybaseUnwiredPlatformRSOE<*install_number*>
-

Unwired Server Performance Counter Reference

This topic covers counters used in current logging and tracing done by Unwired Server.

The counters are by default logged in the log file `<SUP_HOME>/logs/counters.log`. This can be changed by modifying `<SUP_HOME>/config/log4j.properties`. The general format of counters is:

```
<timestamp> | <thread-id> | <counter-name> : <counter-specific-info>
```

Unwired Server generates a trace log message at specific points during processing, called counters. Some of the counters list time spent at various stages of activity. The counters logs can be post-processed to generate aggregate numbers also. The counters can be enabled or disabled in `sup.properties`. The property names are `counter.<counter-name>`, for example, `counter.sync.time`.

You enable counters in the `sup.properties` file. The following counters are most likely to be useful in troubleshooting:

- **sync.time**: Complete synchronization. Note that this is the time spent inside the Unwired Server. The ML Server may add some overhead. This counter prints two lines. At the beginning of sync the following line is printed:

- **sync.time**+:*queue-size.remote-id*

```
2009-04-13 20:09:40,234|Thread-52|sync.time++:
1:df28289b-2275-490b-9e0a-fd524d0ff47f
```

At the beginning of a sync:

- *queue-size* is the total number of sync requests in progress, including the current one.
- **sync.time**:::*queue-size.time-spent.record-count.char-count*

```
2009-04-13 20:09:42,625|Thread-52|sync.time--:0:2391:40:520
```

At the end of the sync:

- *queue-size* is the total number of sync requests in progress, excluding the current one.
- *time-spent* is the time in milliseconds spent in processing the request.
- *record-count* and *char-count* are approximate number of records and characters downloaded to the device respectively. Note that these numbers are inaccurate and are only good for comparison with other lines printed by the same counter.
- **sync.download**: Writing data to download cursors. One line is printed:
 - **sync.download**:*time-spent.record-count.char-count*

```
sync.download:31:40:520
```

- **handle.download.data:** Download phase of MobiLink sync. Unwired Server in this phase does EIS interaction for MBOs (but not operations), CDB Update, and writes to download cursors. The counter prints several lines, but only three are relevant:

- **handle_download_data:+**

```
2009-04-13 20:09:40,328|Thread-52|handle_download_data:+
```

Beginning of download phase.

- **handle_download_data:m:free-memory**

```
2009-04-13 20:09:40,328|Thread-52|handle_download_data:m:
250642128
```

Also at the beginning of download phase; free memory is in bytes.

- **handle_download_data:-:time-spent**

```
2009-04-13 20:09:42,578|Thread-52|handle_download_data:-:2250
```

- **uascripts.getPlaybackTree:** Processing for each top level MBO request, includes EIS Interaction and CDB Update. Two lines are printed, one at the beginning and one at the end.

- **uascripts.getPlaybackTree:+**

```
2009-04-13 20:09:40,421|Thread-52|uascripts.getPlaybackTree:+
```

Beginning of download phase.

- **uascripts.getPlaybackTree:-:time-spent**

```
2009-04-13 20:09:42,406|Thread-52|uascripts.getPlaybackTree:-:
1985
```

- **push.init:** Initializing data structures for push processing. Several lines are printed, but only the beginning and end lines are important.
- **push.process:** Initializing data structures for push processing. Several lines are printed, but only the beginning and end lines are important
- **lock.wait:** While a single property in the `sup.properties` file is provided, the counter names used are `cdb_lock.read` for reading cdb, `playback_lock.write` for doing a playback, and `cdb_lock.write` for updating cdb. Prints one line each when lock is requested and acquired.

```
2009-04-13 20:09:41,062|Thread-52|cdb_lock.read.wait:+
2009-04-13 20:09:41,062|Thread-52|cdb_lock.read.wait:-:0
2009-04-13 20:09:41,078|Thread-52|playback_lock.write.wait:+
2009-04-13 20:09:41,078|Thread-52|playback_lock.write.wait:-:0
2009-04-13 20:09:41,640|Thread-52|cdb_lock.write.wait:+
2009-04-13 20:09:41,640|Thread-52|cdb_lock.write.wait:-:0
```

- **lock.held:** A single property listing the time for which the lock was held. A single line is printed when releasing the lock. The end of `lock.wait` is the beginning of `lock.held`.

```
2009-04-13 20:09:42,390|Thread-52|cdb_lock.held:-:750
2009-04-13 20:09:42,390|Thread-52|playback_lock.held:-:1312
```


- **eis.time:** Two lines are printed, one at the beginning and one at the end of every EIS Interaction, both for MBO read and operation replay.

Troubleshooting

This troubleshooting information is new for Sybase Unwired Platform version 1.2.1.

Changing Service Names

Problem: If the `ClusterRegistration` utility cannot connect to the cluster database when you install and configure Unwired Server, an incorrect service name may be created for the server.

The service name should be "SybaseUnwiredPlatform<cluster>Server<n>," where *cluster* is the name of the cluster, and *n* indicates the order in which the server was added to the cluster.

Solution:

1. Verify that the cluster database (clusterdb) is running.
2. Change to `<Install_Dir>\UnwiredPlatform-1_2\Servers\UnwiredServer`.
3. To remove the service name, open a command window, and run:

mlservice remove

4. To uninstall the service, run:

updateProps -x

5. To reregister the service, run:

updateProps -r

An appropriate value is assigned to the `sup.install.number` property.

6. To re-create the service with the correct name, run:

mlservice install

Cluster and Relay Server Issues

Improve the performance of clusters and relay servers, and troubleshoot known issues.

Cluster Naming Conventions

Follow these cluster naming conventions.

During installation of Sybase Unwired Platform (SUP), if Unwired Server is to be installed as a node in a cluster, you are given the option to change the default name of the cluster (assuming this is the first node of the cluster). The default name is the computer name on which you are installing.

If the cluster is to be part of a hosted Relay Server, use only alphanumeric names, since Relay Server does not allow underbars, spaces, hyphens, and so on.

Each Unwired Server that joins the cluster assumes the name of the cluster incrementally. For example, the first server is named SybaseUnwiredPlatformMYHOSTServer1, the second SybaseUnwiredPlatformMYHOSTServer2, and so on.

Synchronizing an Unwired Server Cluster

Use HTTP to synchronize Unwired Server nodes within a cluster.

Description: Using HTTPS to synchronize Unwired Server nodes in a cluster is ineffective because the Relay Server Outbound Enabler (RSOE), which is used by Unwired Server to communicate with a relay server, does not support HTTPS.

Solution: Verify that each Unwired Server in a cluster uses HTTP as the synchronization protocol, which is the default. Since clusters (including RSOEs) exist behind a firewall, they use the firewall's network security. For each server:

1. In Sybase Control Center, expand Unwired Server.
2. Select **Server Configuration**.
3. Select the **Synchronization** tab.
4. Verify **HTTP** is selected.

Setting the Recovery Level for the Relay Server Outbound Enabler

Use Windows administrative tools to set the the recovery level of the Relay Server Outbound Enabler (RSOE) service to automatically start after a failure.

Description: In some cases the RSOE service may time out due to unforeseen issues (network latency for example) before it connects to Relay Server.

Solution: For each RSOE, configure the RSOE service so it automatically restarts after any failure.

1. Assuming that Relay Server is installed as a Windows Service, from Windows select **Start > Settings > Control Panel**.
2. Select **Administrative Tools > Services**.
3. Double-click the RSOE service (for example, **SybaseUnwiredPlatformclusternameRSOE1**, where *clustername* is the name of the cluster).
4. Select the **Recovery** tab, and select **Restart the Service** for each failure level:
 - First failure
 - Second failure
 - Subsequent failures
5. Set the **Restart service after** level to one minute.

Cluster Synchronization Exceptions

Cluster synchronization exceptions are reported after shutting down the primary Unwired Server.

Problem: This typically happens after shutting down the primary Unwired Server, soon after performing a cluster-changing operation [such as updating the Schedule property for a Mobile Business Object (MBO)]. The shut down occurs before any of the secondary servers received the updated `clusterdata.zip` file. The secondary servers detect the version difference in the `clusterdata.zip` files, shuts down RSOE, and issues an HTTP GET to download the newer file version. Since the primary server was shutdown, the file cannot be downloaded, a secondary cluster cannot become the primary, and cluster sync exceptions are reported in the `ML.log` file in this format:

```
ML.log
=====
E. 2009-06-22 10:09:56. < Main > [-10133] SEVERE: This SUP cluster
member is
at version 14 while the cluster is at version 15
I. 2009-06-22 10:09:59. < Main > The primary server in the server
farm is now:
'GOLDENGATESUPServer3'
I. 2009-06-22 10:09:59. < Main > < SISI > < SISManager > : This is a
secondary
MobiLink server
E. 2009-06-22 10:10:00. < Main > [-10133] Jun 22, 2009 10:10:00 AM
com.sybase.ep.utils.ClusterUtils$1 invoke
E. 2009-06-22 10:10:00. < Main > [-10133] SEVERE: This SUP cluster
member is
at version 14 while the cluster is at version 15
E. 2009-06-22 10:10:00. < Main > [-10133] Jun 22, 2009 10:10:00 AM
com.sybase.ep.utils.ClusterUtils isVersionCorrect
```

Workaround: If the primary Unwired Server cannot be restarted, you can fix the broken cluster manually:

1. Shut down the primary Unwired Server.
2. Copy across the `clusterdata.<version>.zip`.
3. Edit `sup.properties` manually to set `cluster.version` and `sup.cluster.version` to the new value (for this example, this value would be 15).
4. Run **updateProps -r** to load the new version information into the clusterdb.
5. Run **updateFiles** to unzip the `clusterdata.zip`.
6. Start Unwired Server. Since its version now matches the cluster version in clusterdb, it will take over as the primary server.

Any other secondary servers that are still trying to sync will detect the new primary server and will initiate the HTTP GET for the `clusterdata.zip` from the new primary server.

Troubleshooting Communication Errors Between the Client Application and Unwired Server

Add `COMMUNICATION_ERROR_CODE` code to your client applications to gather communication error details.

While the Unwired Server `m1.log` file logs communication errors when there are synchronization errors between Unwired Server and the client, adding `COMMUNICATION_ERROR_CODE` code to the client application provides more detailed information about the types of errors encountered (unable to locate host, timeout, authentication) and is useful in isolating the problem, as this code example illustrates:

```

public void Sync()
{
    try
    {
        Product.Synchronize();
        MessageBox.Show("Sync Done");
    }
    catch (SUPClientException ex)
    {
        if (ex.ErrorCode ==
SUPClientException.COMMUNICATION_ERROR)
        {
            COMMUNICATION_ERROR_CODE code =
ex.CommunicationErrorCode;
            if (code ==
COMMUNICATION_ERROR_CODE.SOCKET_HOST_NAME_NOT_FOUND)
            {
                MessageBox.Show("Server not found. Check host
name.");
            }
            else if (code ==
COMMUNICATION_ERROR_CODE.CONNECT_TIMEOUT)
            {
                MessageBox.Show("Connection timed out. Try again
later.");
            }
            else if (code ==
COMMUNICATION_ERROR_CODE.AUTHENTICATION_FAILED)
            {
                MessageBox.Show("Authentication failed. Check
credentials.");
            }
            else
            {
                MessageBox.Show(code.ToString());
            }
        }
        else
        {
            MessageBox.Show(ex.ToString());
        }
    }
}

```

```

        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString());
        }
    }

```

Sybase Control Center Issues Retrieving Server List

Problem: Sybase Control Center is not functioning because it cannot retrieve its list of Unwired Servers.

Solution: Check the `agent.log` for any related error messages. If there are no related errors, or several 404 errors, this indicates the error occurred outside of the SCC administration console. If there is a related error, it may indicate a caching error. To clear the cache:

1. Delete the contents of:

```

<install_dir>\Sybase\UAF-2_6\services\EmbeddedWebContainer
\container\Jetty-6.1.6\work

```

2. Restart the UAF service.

Unwired Server Reports ODBC Connection Problems

Performance tuning may be required to correct Unwired Server ODBC connection errors.

Problem: If Unwired Server reports ODBC errors similar to the following, this means Unwired Server cannot communicate with the consolidated database:

```

I. 2009-05-06 00:00:17. Old output file "C:\www\Sybase
\UnwiredPlatform-1_2\Servers\UnwiredServer\logs\ml.log"
has been renamed to file "C:\www\Sybase\UnwiredPlatform-1_2\Servers
\UnwiredServer\logs\ml.log.old"
E. 2009-05-06 00:00:17. < 7382 > [-10279] Connection was dropped due
to lack of network activity
E. 2009-05-06 00:05:18. < 7388 > [-10279] Connection was dropped due
to lack of network activity
E. 2009-05-06 00:10:18. < 7394 > [-10279] Connection was dropped due
to lack of network activity
E. 2009-05-06 00:12:10. < Main > [-10002] Consolidated database
server or ODBC error: ODBC: [Sybase][ODBC Driver][SQL Anywhere]
Connection was terminated (ODBC State = HY000, Native error code =
-308)
E. 2009-05-06 00:12:10. < Main > [-10002] Consolidated database
server or ODBC error: ODBC: [Sybase][ODBC Driver][SQL Anywhere]
Connection was terminated (ODBC State = HY000, Native error code =
-308)
I. 2009-05-06 00:12:10. < Main > java.sql.SQLException: [Sybase]
[ODBC Driver][SQL Anywhere]Connection was terminated

```

Solution: Try changing the setting in the `ml_property` table to a higher value. For example, changing the property from the default 60 (seconds) to 150 lengthens the window of

opportunity for Unwired Server to communicate with the consolidated database, and should correct the ODBC connection error.

1. See whether the property row exists, using this dbisql query:

```
SELECT * FROM ml_property WHERE property_name = 'liveness'
```

If nothing returns, then the row does not exist.

2. Take action based on whether the row exists.

- If the row exists, the property has been set. Use the following stored procedure to set a higher value:

```
exec ml_add_property 'MLS', 'ml_server_farm',  
'liveness', '150' ;
```

- If the row does not exist, the property has not been set. Use the following INSERT statement to set the property, which adds the row to the table:

```
Begin  
INSERT INTO ml_property ( component_name, property_set_name,  
property_name, property_value)  
VALUES( 'MLS', 'ml_server_farm', 'liveness', '150')  
End  
COMMIT
```

Resolving Client-Side Certificate Errors

Problem: A remote client does not run when using a client-side certificate, and an error is reported, such as -403 HTTP or -413 HTTP.

Workaround:

1. Take action based on the error reported:

- -403 HTTP error – the IIS server reports this Unauthorised Error if you do not supply the identity and identity_password correctly.

Verify that you provided the correct values for the Stream parameter as described in *Running the Client* on page 114.

- - 413 HTTP error – the IIS server may not have the entire certificate.

Try running this script:

```
cscript adsutil.vbs set w3svc/1/uploadreadaheadsize 65536
```

- StreamErrorCode=SECURE_TRUSTED_CERTIFICATE_FILE_NOT_FOUND.
Verify that the certificate files have been correctly copied to the emulator or device, and that the value of Output_File_Folder is correct.
- StreamErrorCode=SECURE_CERTIFICATE_NOT_TRUSTED.
Verify that the certificate files are correct. If they are correct, the IIS server may not be configured correctly for client-side certificate authentication.
- Other errors – these may indicate the IIS server is not configured correctly for client-side certificate authentication.

Check the IIS configuration as described in *Setting Up RS_Client for Client-Side Certificate Authentication* on page 113.

2. To learn more about Microsoft IIS-related errors, see:

[http://technet.microsoft.com/en-us/library/cc737382\(WS.10\).aspx](http://technet.microsoft.com/en-us/library/cc737382(WS.10).aspx)

Inserting Data That Exceeds Maximum Size Causes Errors

Problem: If you insert or update data whose size is greater than the maximum size defined for the column, an error occurs.

The following table describes the errors that occur when the data size exceeds the column's maximum size (`maxLength`):

Datatype	Column maximum size	If the data size exceeds <code>maxLength</code> , the result is
Binary	Less than 2K	Data is silently truncated in the field
Binary	Greater than 2K and less than 32K	MobiLink server throws an exception; you see a <code>transaction failure</code> message in <code>ml.log</code>
Binary	Greater than 32K	You receive a synchronization exception; parameter length mismatch error in <code>SyncResult</code>
String	Less than 8K	Data is silently truncated in the field
String	Greater than 8K	You receive a synchronization exception; parameter length mismatch error in <code>SyncResult</code>

Workaround: Use the client API to verify that the size of data entered by users is within limits. See the *Client Object API Cookbook*, available on Sybase Unwired Platform Tech Corner at <http://www.sybase.com/developer/library/suptechcorner>.

Index

A

- actions
 - BlackBerry PIM action 56
- admin api 1
- alternate read
 - covering playback and sync parameters 35
 - requirements 34
- API
 - <MBO>PendingState 11
 - evict 11
 - FindByOperationID 11
 - GetDataCount (Query query) 11
 - isPendingOverwritten 11
 - Load[relationship_name]
 - (List<parent_mbo_name>) 11
- apply operation parameters cache policy 77
- apply operation result cache policy 76
- AutoCommit feature, SAP 85
- autocommitPreviewTransaction property 65

B

- backup
 - consolidated database 13, 16
 - installation file system 12
 - transaction log 13, 16
- backup and recovery plan 14
- BlackBerry PIM action, adding 56

C

- cache update policies
 - apply operation parameters 77
 - apply operation result 76
 - chained operation 78
 - defined 25, 74, 117
 - invalidate 29, 82
 - no invalidate 29, 82
 - setting 75
 - validation rules 83
- cascade delete
 - admin API 1
 - deployment default 1

- deployment unit 1
- MBO relationships 1
- certificates
 - client-side, errors 138
 - generating for client-side synchronization 109
 - generating for HTTPS synchronization 98, 125
- chained operation cache policy 78
- chained operations
 - defining 80
 - playback and sync parameters 81
 - requirements 81
- changing service names 133
- choice control properties 54
- client-side certificate errors, resolving 138
- client-side certificates 116
- clusterdb.db 14
- clusterdb.log file 14
- clusters
 - synchronizing 134
- composition relationship 1
- configure Microsoft IIS 109
- configuring
 - SAP AutoCommit feature 85
 - SAP environment 45, 99, 126
- configuring logs on separate drives 15
- connection properties
 - SAP, modifying 98
- consolidated database
 - backup 13, 16
 - clusterdb.db 14
 - dedicated for a cluster 106
 - restore 17
 - setting the thread count manually 130
 - uaml.db 14
- covering playback and sync parameters 35
- crash and recovery scenarios 14
- createcert, command line utility 98, 109, 125

D

- data change notification
 - filters 9, 121
- data change notification filter
 - example 10

- implementing 10
- data change notification interface 2
- data change notification parameters 6
- data change notification syntax 6
- data change notification with payload 6
- data change notification, results 8
- dbbackup utility 14, 16
- dblocate utility 16
- dbvalid utility 16, 17
- deploying
 - SAP result checker classes 39, 72
- deployment unit 1
- Device Application Designer
 - automatic screen creation 50
 - creating 50
 - device application screens 50
 - documentation updates 48
- device database page size 101
- disaster recovery planning 12
- documentation updates
 - Unwired WorkSpace Visual Studio Edition 69

E

- errors
 - data that exceeds size limits 139
- exceeding maximum size errors 139

F

- file system
 - backup 12
 - restore 17
- filters
 - data change notification 9, 121

G

- generating code
 - troubleshooting slow performance 102
- GetCalledOperations method 101
- GUI is not generated for EIS connection properties
 - troubleshooting 66

H

- HTTP interface for data change notification 2
- HTTPS

- generating certificates for 98, 125
- generating certificates for client-side 109

I

- installation file system
 - backup 12
 - restore 17
- invalidate cache policy 29, 82

L

- LDAP provider
 - Sybase Control Center 127
- linked parameters 86
 - transforming a data source for 87
- linking SAP parameters to arguments 102

M

- mobile business objects
 - attribute and parameter size restrictions 101
 - troubleshooting synchronization errors 67, 103
- mobile device client recovery 12, 13, 16, 17
- multilevel insert operations
 - creating for Web-service MBOs 22, 94

N

- new features
 - Unwired WorkSpace Eclipse Edition 19
 - Unwired WorkSpace Visual Studio Edition 69
- no invalidate cache policy 29, 82

P

- parameters
 - linked 86
 - playback and sync for chained operations 81
- parameters, data change notification
 - dcn_request 6
 - domain 6
 - package 6
 - password 6
 - unwired_server 6
 - unwired_server_port 6

- username 6
- performing remote backup 16
- playback and synchronization parameters 81
- policies
 - apply operation parameters for cache updates 77
 - apply operation results to cache 76
 - cache updates, defining 75
 - chained operation cache update 78
 - invalidate cache 29, 82
 - no invalidate cache 29, 82
- previews
 - trouble extracting metadata 65
- properties
 - choice control 54

R

- refactoring an SAP result checker 40, 73
- reference relationship 1
- registering a service 133
- relationship, composition 1
- relationship, reference 1
- Relay Server 109
- remote backup 16
- removing a service 133
- restore
 - consolidated database 17
 - installation file system 17
 - transaction log 17
 - Windows registry 17
- result set filter, creating 64

S

- SAP
 - AutoCommit feature, configuring 85
 - configuring Unwired Platform components for 45, 99, 126
 - connection properties, modifying 98
 - operations, committing 41
 - parameter-to-argument linking 102
 - result checker classes, deploying 39, 72
 - result checker, customizing 38, 71
- SAP connection errors
 - missing sapmsg.ini file 102
- SAP result checkers
 - configuring 73

- creating 71
- deleting 40, 73
- moving 40, 73
- references, finding 40, 74
- renaming 40, 73
- sapmsg.ini file is missing 102
- services
 - changing the name 133
- SQLLE_TOO_MANY_PUBLICATIONS error,
 - troubleshooting 66, 102
- Sybase Control Center
 - configuring server properties 122
 - LDAP provider 127
 - new features 103
- synchronization
 - exception, data size exceeds maxlen 139
 - troubleshooting 67, 103
- synchronizing
 - cluster 134

T

- thread count, setting for the CDB 130
- transaction failure message 139
- transaction log
 - backup 13, 16
 - clusterdb.log 14
 - configuring logs on separate drives 15
 - configuring mirrored logs on separate drives 15
 - restore 17
 - uaml.log 14
- transforming a data source 87
- troubleshooting
 - BlackBerry device applications 67
 - cannot preview data 65
 - client-side certificate errors 138
 - Codegen.GetCalledOperations 101
 - GUI is not generated for EIS connection
 - properties as parameters 66
 - SAP connection errors 102
 - slow code generation, Visual Studio 102
 - SQLLE_TOO_MANY_PUBLICATIONS error 66, 102
 - synchronization 67, 103
 - Visual Studio 2008 Chinese Edition 103
 - Windows Mobile device applications 67
- troubleshooting topics
 - Unwired WorkSpace Visual Studio Edition 69

U

uaml.db database file 14

uaml.log file 14

Unwired Server

new features 103

Unwired WorkSpace

Eclipse Edition, new features 19

Visual Studio Edition, new features 69

utilities

dbbackup 14, 16

dblocate 16

dbvalid 16, 17

V

validation rules for cache update policies 83

Visual Studio

Chinese Edition, troubleshooting 103

W

Web service MBOs

deployment fails, troubleshooting 65

multilevel insert operations, creating for 22,
94

Windows registry

restore 17

writing a custom result set filter, correction 64