SYBASE®
An SAP® Company

## New Features

# Sybase Unwired Platform 2.0 ESD #1

DOCUMENT ID: DC01203-01-0201-01
LAST REVISED: July 2011

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

# Contents

# New or Updated Features

Sybase® Unwired Platform covers the end-to-end aspects of mobilizing an application—providing standards-based access to data sources, offering consistent and reusable object-oriented frameworks across heterogenous device platforms, supporting end-to-end security, and managing deployment, devices, users, and applications. Sybase Unwired Platform includes several important new or updated features described here.

## Sybase Mobile Workflow

The Sybase Mobile Workflow container is now available for Android with support equivalent to that available for iOS, BlackBerry, and Windows Mobile.

Supported Android versions:

- 2.2 and higher
- 3.0

Documented in:

- *Developer Guide for Mobile Workflow Packages*
- *Fundamentals*
- *Tutorial: Mobile Workflow Package Development*

## Object Query Enhancements for Windows Mobile

SQL features and functions have been added to the dynamic query API.

The Object API provides increased access to the underlying relational persistence layer. SQL features and functions have been added to the dynamic query method, `ExecuteQuery`. The Object API is documented in the *Developer Guide for Windows and Windows Mobile*.

The additions are intended to permit developers to:

- (Update) Exclude duplicate entries from result sets using the arbitrary find method `Query.DISTINCT` property.
- (Update) Define a filter condition using an MBO attribute using the new conditions, IN, NOT_IN, EXISTS, and NOT_EXISTS. You can nest queries, as well as use `LIKE` keywords in queries.
- (New) Use aggregate functions.
- (New) Group your results using specific attributes.
- (New) Specify how your results are filtered.

- (New) Combine the results of multiple queries into a single result set.
- (New) Include subqueries in `SELECT` statements.

## Arbitrary Find

The arbitrary find method lets custom device applications dynamically build queries based on user input. The `Query.DISTINCT` property has been added so you can exclude duplicate entries from the result set. This information supersedes the existing information in the *Developer Guide for Windows and Windows Mobile*.

The arbitrary find method also lets the user specify a desired ordering of the results and object state criteria. A `Query` class is included in the client object API. The `Query` class is the single object passed to the arbitrary search methods and consists of search conditions, object/row state filter conditions, and data ordering information.

Define these conditions by setting properties in a query:

- **TestCriteria –** criteria used to filter returned data.
- **SortCriteria –** criteria used to order returned data.
- **Skip –** an integer specifying how many rows to skip. Used for paging.
- **Take –** an integer specifying the maximum number of rows to return. Used for paging.

Set the `Query.DISTINCT` property to `true` to exclude duplicate entries from the result set. The default value is `false` for entity types, and its usage is optional for all other types.

```
Query query1 = new Query();
query1.DISTINCT = true;
```

`TestCriteria` can be an `AttributeTest` or a `CompositeTest`.

## AttributeTest

An `AttributeTest` defines a filter condition using an MBO attribute, and supports multiple conditions. The IN, NOT_IN, EXISTS, and NOT_EXISTS conditions have been added. This information supersedes the existing information in the *Developer Guide for Windows and Windows Mobile*.

- IS_NULL
- NOT_NULL
- EQUAL
- NOT_EQUAL
- LIKE
- NOT_LIKE
- LESS_THAN
- LESS_EQUAL
- GREATER_THAN
- GREATER_EQUAL

- CONTAINS
- STARTS_WITH
- ENDS_WITH
- DOES_NOT_START_WITH
- DOES_NOT_END_WITH
- DOES_NOT_CONTAIN
- IN
- NOT_IN
- EXISTS
- NOT_EXISTS

For example, the C# .NET code shown below is equivalent to this SQL query:

```
SELECT a.id from AllType a where a.id in (1,2,3)
```

```
Sybase.Persistence.Query query = new Sybase.Persistence.Query();
query.Select("a.id");
query.From("AllType", "a");
Sybase.Persistence.AttributeTest ts2 = new
Sybase.Persistence.AttributeTest();
ts2.Attribute = "a.id";

Sybase.Collections.ObjectList v = new
Sybase.Collections.ObjectList();
v.Add(1);
v.Add(2);
v.Add(3);
ts2.Value = v;
ts2.SetOperator(Sybase.Persistence.AttributeTest.IN);
query.Where(ts2);
Sybase.Persistence.QueryResultSet qs = DsTestDB.ExecuteQuery(query);
```

When using EXISTS and NOT_EXISTS, the attribute name is not required in the `AttributeTest`. The query can reference an attribute value via its alias in the outer scope. The C# .NET code shown below is equivalent to this SQL query:

```
SELECT a.id from AllType a where exists (select b.id from AllType b
where b.id = a.id)
```

```
Sybase.Persistence.Query query = new Sybase.Persistence.Query();
query.Select("a.id");
query.From("AllType", "a");
Sybase.Persistence.AttributeTest test = new
Sybase.Persistence.AttributeTest();
Sybase.Persistence.Query existQuery = new
Sybase.Persistence.Query();
existQuery.Select("b.id");
existQuery.From("AllType", "b");
Sybase.Persistence.Column cl = new Sybase.Persistence.Column();
cl.Alias = "a";
cl.Attribute = "id";
Sybase.Persistence.AttributeTest test1 = new
Sybase.Persistence.AttributeTest();
test1.Attribute = "b.id";
```

```
test1.Value = cl;
test1.SetOperator(Sybase.Persistence.AttributeTest.EQUAL);
existQuery.Where(test1);
test.Value = existQuery;
test.SetOperator(Sybase.Persistence.AttributeTest.EXISTS);
query.Where(test);
Sybase.Persistence.QueryResultSet qs = DsTestDB.ExecuteQuery(query);
```

## Aggregate Functions

You can use aggregate functions in dynamic queries.

When using the `Query.Select(String)` method, you can use any of these aggregate functions:

| Aggregate Function | Supported Datatypes |
|---|---|
| `COUNT` | integer |
| `MAX` | string, binary, char, byte, short, int, long, integer, decimal, float, double, date, time, dateTime |
| `MIN` | string, binary, char, byte, short, int, long, integer, decimal, float, double, date, time, dateTime |
| `SUM` | byte, short, int, long, integer, decimal, float, double |
| `AVG` | byte, short, int, long, integer, decimal, float, double |

If an unsupported type is used, a `PersistenceException` is thrown.

```
Query query1 = new Query();
query1.Select("MAX(c.id), MIN(c.name) as minName");
```

## Grouping Results

Apply grouping criteria to your results.

To group your results according to specific attributes, use the `Query.groupBy(String groupByItem)` method. For example, to group your results by ID and name, use:

```
String groupByItem = ("c.id, c.name");
Query query1 = new Query();

//other code for query1

query1.GroupBy(groupByItem);
```

## Filtering Results

Specify test criteria for dynamic queries.

You can specify how your results are filtered by using the `Query.having(com.sybase.persistence.TestCriteria)` method. For example, limit your results to `id` attribute values that are greater than or equal to 0 using:

```
Query query1 = new Query();
//other code for query1

AttributeTest ts2 = new AttributeTest();
ts2.Attribute = "id";
ts2.Value = "0";
ts2.SetOperator(AttributeTest.GREATER_EQUAL);
query1.Having(ts2);
```

## Concatenating Queries

Concatenate two queries having the same selected items.

The `Query` class methods for concatenating queries are:

- `Union(Query)`
- `UnionAll(Query)`
- `Except(Query)`
- `Intersect(Query)`

This example obtains the results from one query except for those results appearing in a second query:

```
Query query1 = new Query();
... ... //other code for query1

Query query2 = new Query();
... ... //other code for query 2

Query query3 = query1.Except(query2);
SampleAppDB.ExecuteQuery(query3);
```

## Subqueries

Execute subqueries using clauses, selected items, and attribute test values.

You can execute subqueries using the `Query.from(Query query, String alias)` method. For example, the C# .NET code shown below is equivalent to this SQL query:

```
SELECT a.id FROM (SELECT b.id FROM AllType b) AS a WHERE a.id = 1
```

Use:

```
Query query1 = new Query();
query1.Select("b.id");
query1.From("AllType", "b");
Query query2 = new Query();
```

```
query2.Select("a.id");
query2.From(query1, "a");
AttributeTest ts = new AttributeTest();
ts.Attribute = "a.id";
ts.Value = 1;
query2.Where(ts);
Sybase.Persistence.QueryResultSet qs =
DsTestDB.ExecuteQuery(query2);
```

You can use a subquery as the selected item of a query. Use the `SelectItem` to set selected items directly. For example, the C# .NET code shown below is equivalent to this SQL query:

```
SELECT (SELECT count(1) FROM AllType c WHERE c.id >= d.id) AS cn, id
FROM AllType d
```

Use:

```
Query selQuery = new Query();
selQuery.Select("count(1)");
selQuery.From("AllType", "c");
AttributeTest ttt = new AttributeTest();
ttt.Attribute = "c.id";
ttt.SetOperator(AttributeTest.GREATER_EQUAL);
Column cl = new Column();
cl.Alias = "d";
cl.Attribute = "id";
ttt.Value = cl;
selQuery.Where(ttt);

Sybase.Collections.GenericList<Sybase.Persistence.SelectItem>
selectItems = new
Sybase.Collections.GenericList<Sybase.Persistence.SelectItem>();
SelectItem item = new SelectItem();
item.Query = selQuery;
item.AsAlias = "cn";
selectItems.Add(item);
item = new SelectItem();
item.Attribute = "id";
item.Alias = "d";
selectItems.Add(item);
Query subQuery2 = new Query();
subQuery2.SelectItems = selectItems;
subQuery2.From("AllType", "d");
Sybase.Persistence.QueryResultSet qs =
DsTestDB.ExecuteQuery(subQuery2);
```

## Result Checker Logging

Use OHLog to trap warnings but not halt execution of the result checker.

(New) You can influence the error or warning code and message in the result checker by throwing a `DSException`, which produces errors and halts execution, or by calling `OHLog`, which is used for warnings and does not halt execution.

Use OHLog.log() to write to the client log. This method returns `true` if it successfully wrote the log entry, and `false` if no client is defined. For example, no client is typically defined for a scheduled refresh.

*Data Source: SAP*

This code sample indicates how to use `OHLog` for an SAP® back end.

```
package com.sybase.vader.test.mms;

import java.util.AbstractMap;
import java.util.Map;

import com.sap.mw.jco.JCO;
import com.sybase.sup.sap.SAPResultChecker;
import com.sybase.dataservices.DSException;
import com.sybase.dataservices.OHLog;

public class TestSAPResultChecker implements SAPResultChecker {
        public Map.Entry<Boolean, String> checkReturn( JCO.Function
         f ) {
        JCO.Record returnStructure = null;
        JCO.ParameterList jpl = f.getExportParameterList();
        int supCode = 200;  // Use a standard http code, or 900+
         for custom
        int eisCode = 0; // Use code returned by backend system
        if ( jpl != null )
        {
            try
            {
                returnStructure = jpl.getStructure("RETURN");
                if ( returnStructure != null )
                {
                    String type = returnStructure.getString("TYPE");
                    String message =
                     returnStructure.getString("MESSAGE");
                    eisCode = returnStructure.getInt("NUMBER");
                    OHLog.log(supCode, eisCode, "TYPE: " + type,
                     OHLog.DEBUG);
                    if ( !(type.equals("") || type.equals("S") ||
                     type.equals("I")) )
                    {
                        if ( !type.equals("W"))
                        {
                            throw new DSException
                             (DSException.INTERNAL_SERVER_ERROR,
                             eisCode, message);
                        }
                        else
                        {
                            OHLog.log(supCode, eisCode, message,
                             OHLog.WARN);
                        }
                    }
                }
```

```
            }
            catch (DSException dse) {
                throw dse;
            }
            catch (Exception e) {
                OHLog.log(OHLog.EIS_RESOURCE_NOT_FOUND, 0,
              e.getMessage(),
              OHLog.WARN);
            }
        }
        else {
            OHLog.log(200, 0, "No parameter list returned",
          OHLog.WARN);
        }
               return new AbstractMap.SimpleEntry<Boolean,
          String>( true, "" );
        }
}
```

*Data Source: Web Service (SOAP)*

This code sample indicates how to use `OHLog` for a SOAP Web service back end.

```
package com.sybase.vader.test.mms;

import java.io.StringWriter;
import java.util.AbstractMap;import java.util.Map.Entry;

import com.sybase.dataservices.DSException;
import com.sybase.dataservices.OHLog;
import com.sybase.sup.ws.soap.SoapOperationHandler;
import com.sybase.sup.ws.soap.WSResultChecker;

import javax.xml.transform.OutputKeys;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.Transformer;

import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import javax.xml.soap.SOAPFault;
import javax.xml.soap.SOAPEnvelope;

public class TestSoapResultChecker implements WSResultChecker {
   public Entry<Boolean, String> checkReturn(SOAPEnvelope response)
{
 int supCode = 900;  // Use a standard http code, or 900+ for custom
 int eisCode = 0; // Use code returned by backend system
    OHLog.log(supCode, eisCode, toXML(response), OHLog.DEBUG);

               try{
               SOAPFault fault = response.getBody().getFault();
               if(fault!=null) {
                       throw new DSException
            (DSException.INTERNAL_SERVER_ERROR,
             Integer.valueOf(fault.getFaultCode()),
```

```
              fault.getFaultString());
                }
          }
          catch (DSException dse) {
                    throw dse;
          }        catch (Exception e) {
       OHLog.log(OHLog.EIS_RESOURCE_NOT_FOUND, 0, e.getMessage(),
        OHLog.WARN);
          }
     return  new AbstractMap.SimpleEntry<Boolean, String>
   ( true, "" );
    }

 private String toXML(javax.xml.soap.SOAPEnvelope env) {
               String xmlString="";
               try {
                      TransformerFactory transfac =
              TransformerFactory.newInstance();
                      Transformer trans = transfac.newTransformer();
                      trans.setOutputProperty
              (OutputKeys.OMIT_XML_DECLARATION, "yes");
                      trans.setOutputProperty(OutputKeys.INDENT,
               "yes");

                      StringWriter sw = new StringWriter();
                      StreamResult result = new StreamResult(sw);
                      DOMSource source = new DOMSource(env);
                      trans.transform(source, result);
                      xmlString = sw.toString();
                  }
               catch(Exception e) {
                     }
               return xmlString;
  }
}
```

*Data Source: RESTful Web Service*

This code sample indicates how to use `OHLog` for a RESTful Web service back end.

```
package com.sybase.vader.test.mms;

import java.net.URL;
import java.util.AbstractMap;
import java.util.List;
import java.util.Map;

import com.sybase.sup.ws.rest.RestResultChecker;
import com.sybase.dataservices.OHLog;
import com.sybase.dataservices.DSException;

public class TestRestResultChecker implements RestResultChecker {

    public Map.Entry<Boolean, String> checkReturn( String
  responseBody,
                   List<List<String>> responseHeaders, int
```

```
        httpStatusCode )
    {
        int supCode = 900; // Use a standard http code, or 900+ for
    custom
       int eisCode = httpStatusCode; // Use code returned by backend
    system
      OHLog.log(supCode, eisCode, "httpStatusCode="+httpStatusCode,
    OHLog.INFO);
        if(responseBody != null) {
            if(responseBody.isEmpty()){
                OHLog.log(supCode, eisCode, "response body empty",
     OHLog.WARN);
            }
            else {
                OHLog.log(supCode, eisCode, responseBody,
     OHLog.DEBUG);
            }
        }
        else {
             OHLog.log(supCode, eisCode, "response body null",
     OHLog.WARN);
        }
        int i=1;
        for(List<String> list : responseHeaders) {
            String msg = "" + list.get(0) + "=" + list.get(1);
            OHLog.log(901, i++, msg, OHLog.INFO);
        }               if(httpStatusCode>=300) {
                  throw new DSException
           (DSException.INTERNAL_SERVER_ERROR, httpStatusCode,
          "HTTP status code ["+httpStatusCode+"] too high");
             }
             return new AbstractMap.SimpleEntry<Boolean,
        String>( true, "" );
     }
}
```

# Index

## U



## W