BIRT API Change Control Document

Open Data Access Public Interfaces

Last Updated: September 29, 2005

Abstract

This document tracks the change requests to the public API of the Open Data Access (ODA) framework. For each change request, the document describes the new requirement, proposed solution and follow-up actions.

Document Revisions

Version	Date	Description of Changes
1.1	9/29/2005	Added 2 new methods in the IBlob and IClob interfaces; added description of corresponding support in the oda.consumer.helper in section 1.1
1.0	9/14/2005	Initial Version

1. Introduction

The Open Data Access (ODA) framework defines run-time and design-time interfaces for accessing data from both standard and custom data sources. A data source provider implements such interfaces for consumption by any ODA data consumer applications. The current ODA interfaces version is 2.0.1, released as part of the BIRT project. As described in BPS#30 ODA Framework Migration to the Data Tools Platform (DTP) project (http://www.eclipse.org/birt/wiki/index.php?n=BPS.BPS30), version 2.0.1 is frozen. Any enhancements to ODA will be applied to the DTP ODA version 3.0 or later.

A number of enhancements are being identified for ODA interfaces version 3.0. This API change control document is intended to be a working document that evolves and expands to describe changes in the public interfaces of the ODA framework. As we go through the design phase, more change requests and corresponding proposed changes will be added to the document.

Below sections describe the enhancements and corresponding proposed interface changes in version 3.0, from version 2.0.1.

1.1 Additional data types support of CLOB and BLOB in result set columns – BPS #3 and Bugzilla 95793

http://www.eclipse.org/birt/wiki/index.php?n=BPS.BPS3

https://bugs.eclipse.org/bugs/show_bug.cgi?id=95793

The BLOB and CLOB data types are added in ODA version 3.0 as new ODA scalar data types. The two new data types are supported in an ODA result set column and an ODA output parameter. They are not supported as an ODA input parameter data type.

In order to encapsulate the processing of the CLOB and BLOB data, and to provide room for future extension, a separate Java interface is used for each of these data types. See the Added APIs section below for the new `IBlob` and `IClob` interfaces, and the Changed APIs section for the corresponding getter methods in `IResultSet` and `IAdvancedQuery`.

Note: The `IBlob` and `IClob` interfaces include optional short-cut methods, `IBlob.getBytes` and `IClob.getSubString`, to access all or a portion of the data provided through the binary and character streams. These short-cut methods are intended to facilitate data access by ODA consumer applications that only need to work with binary and string representation of BLOB/CLOB data respectively. An ODA run-time driver may choose to provide an optimized implementation, or simply throws an UnsupportedOperationException. Instead of requiring every underlying ODA driver to support such short-cut methods, the corresponding odaconsumer helper classes in the DTP ODA framework (`OdaBlob` and `OdaClob` in `org.eclipse.datatools.connectivity.oda.consumer.helper` package) provide default implementation to retrieve and return the data through the IBlob's binary stream and IClob's character stream.

An ODA run-time driver that supports the CLOB and/or BLOB data types would provide an implementation that is most efficient for its type of data source. Alternatively, an ODA run-time driver may use the default implementation classes provided by the ODA framework. They are:

```
org.eclipse.datatools.connectivity.oda.impl.Blob
      public Blob( byte[] byteArray )

org.eclipse.datatools.connectivity.oda.impl.Clob
      public Clob( String string )
```

The ODA reference implementation class for each LOB data type interface handles the common type(s) of raw data in a CLOB and BLOB value. For example, a `Blob` constructor takes the argument of a `byte[]` value; and a `Clob` constructor takes the argument of a `String` value.

Note: In some use cases, one might want to associate a BLOB data item with additional attributes, such as hotspot locations. Such association should be mapped in an ODA consumer application such as in a BIRT report item. An ODA run-time driver would thus not be burdened with the implementation of such application-specific usage.

1.2 Pass-through of External Context Objects to ODA Data Providers – BPS #35

http://www.eclipse.org/birt/wiki/index.php?n=BPS.BPS35

An ODA consumer application, such as BIRT, is often embedded as part of a middle-tier application server, where the application components are added to the mix of various J2EE components. During run-time, some of these other components may instantiate context objects, which are served to an ODA data source provider to use.

The DTP ODA run-time public API, in ODA version 3.0, adds new interface methods to allow one to pass a context object into an ODA driver (`IDriver`) instance, plus each of its data source connection (`IConnection`) and data set query (`IQuery`) instances.

A custom ODA run-time driver provided by an application, which embeds the BIRT engine, should implement these ODA interface methods to process the context object, as appropriate.

See the Changed APIs section below on the new interface methods added in `IDriver`, `IConnection` and `IQuery` to pass in a context object.

1.3 Additional enhancements – TBD

To be added as new features are defined.

2. Changed APIs:

2.1 IDriver

Component name = ODA Run-time Interfaces

Package name = org.eclipse.datatools.connectivity.oda

2.1.1 Change Request: Pass-through of External Context Objects to ODA Data Providers.

Proposed Solution:

Add a new setter method in `IDriver` to pass in a context object to the driver instance.

```
/**
 * Sets the driver context.
 * Its handling is specific to individual driver implementation.
 * Note: This method should be called before
 * getConnection( String ).
 * An optional method.
 * If any part of the context is not recognized by the driver,
 * it should simply ignore, and not throw an exception.
 * @param context Context object of this instance.
 * @throws OdaException       if data source error occurs
 * @since         3.0
 */
public void setContext( Object context ) throws OdaException;
```

2.2 IConnection

Component name = ODA Run-time Interfaces

Package name = org.eclipse.datatools.connectivity.oda

2.2.1 Change Request: Pass-through of External Context Objects to ODA Data Providers.

Proposed Solution:

Add a new setter method in `IConnection` to pass in a context object to the connection instance.

```
/**
 * Sets the connection context.
 * Its handling is specific to individual driver implementation.
 * Note: This method should be called before open().
 * An optional method.
 * If any part of the context is not recognized by the driver,
 * it should simply ignore, and not throw an exception.
 * @param context Context object of this instance.
 * @throws OdaException       if data source error occurs
 * @since         3.0
 */
public void setContext( Object context ) throws OdaException;
```

2.3 IQuery

Component name = ODA Run-time Interfaces

Package name = org.eclipse.datatools.connectivity.oda

2.3.1 Change Request: Pass-through of External Context Objects to ODA Data Providers.

Proposed Solution:

Add a new setter method in `IQuery` to pass in a context object to the query instance.

```
/**
 * Sets the query context.
 * Its handling is specific to individual driver implementation.
 * <br>
 * <b>Note:</b> This method should be called before prepare().
 * <br>An optional method.
 * If any part of the context is not recognized by the driver,
 * it should simply ignore, and not throw an exception.
 * @param context Context object of this instance.
 * @throws OdaException       if data source error occurs
 * @since         3.0
 */
public void setContext( Object context ) throws OdaException;
```

2.4 IAdvancedQuery

Component name = ODA Run-time Interfaces

Package name = org.eclipse.datatools.connectivity.oda

2.4.1 Change Request: Support of CLOB and BLOB data types in output parameters.

Proposed Solution:

Add new getter methods in `IAdvancedQuery` to retrieve a parameter's output value as the `IBlob` or `IClob` data type.

```
/**
 * Returns the IBlob value from the designated output parameter.
 * <p><b>Note:</b> The driver must guarantee that
 * the returned IBlob object and its BLOB data would remain valid
 * and accessible until this query instance is closed.
 * @param parameterName       name of the parameter.
 * @return        an IBlob object that represents the BLOB value;
 *                or <code>null</code> if the specific parameter
 *                has null value.
 * @throws OdaException       if data source error occurs
 * @since         3.0
 */
public IBlob getBlob( String parameterName ) throws OdaException;

/**
 * Returns the IBlob value from the designated output parameter.
 * <p><b>Note:</b> The driver must guarantee that
 * the returned IBlob object and its BLOB data would remain valid
 * and accessible until this query instance is closed.
 * @param parameterId         id of the parameter (1-based).
 * @return        an IBlob object that represents the BLOB value;
 *                or <code>null</code> if the specific parameter
 *                has null value.
 * @throws OdaException       if data source error occurs
```

```
     * @since          3.0
     */
    public IBlob getBlob( int parameterId ) throws OdaException;

    /**
     * Returns the IClob value from the designated output parameter.
     * <p><b>Note:</b> The driver must guarantee that
     * the returned IClob object and its CLOB data would remain valid
     * and accessible until this query instance is closed.
     * @param parameterName        name of the parameter.
     * @return         an IClob object that represents the CLOB value;
     *                 or <code>null</code> if the specific parameter
     *                 has null value.
     * @throws OdaException        if data source error occurs
     * @since          3.0
     */
    public IClob getClob( String parameterName ) throws OdaException;

    /**
     * Returns the IClob value from the designated output parameter.
     * <p><b>Note:</b> The driver must guarantee that
     * the returned IClob object and its CLOB data would remain valid
     * and accessible until this query instance is closed.
     * @param parameterId          id of the parameter (1-based).
     * @return         an IClob object that represents the CLOB value;
     *                 or <code>null</code> if the specific parameter
     *                 has null value.
     * @throws OdaException        if data source error occurs
     * @since          3.0
     */
    public IClob getClob( int parameterId ) throws OdaException;
```

2.5 IResultSet

Component name = ODA Run-time Interfaces

Package name = org.eclipse.datatools.connectivity.oda

2.5.1 Change Request: Support of CLOB and BLOB data types in result set columns.

Proposed Solution:

Add new getter methods in `IResultSet` to retrieve a column value as the `IBlob` or `IClob` data type.

```
    /**
     * Gets the value of the designated column in the current row
     * as an IBlob object.
     * Note: The driver must guarantee that
     * the returned object and its BLOB data would remain valid
     * and accessible until this result set is closed.
     * @param index    column number (1-based)
     * @return         an IBlob object that represents the BLOB value
     *                       in the specific column of the current row;
     *                       or <code>null</code> if the specific
     *                       column has null value
     * @throws OdaException        if data source error occurs
     * @since          3.0
     */
    public IBlob getBlob( int index ) throws OdaException;

    /**
```

```
     * Gets the value of the designated column in the current row
     * as an IBlob object.
     * Note: The driver must guarantee that
     * the returned object and its BLOB data would remain valid
     * and accessible until this result set is closed.
     * @param columnName    column name
     * @return        an IBlob object that represents the BLOB value
     *                      in the specific column of the current row;
     *                      or <code>null</code> if the specific
     *                      column has null value
     * @throws OdaException       if data source error occurs
     * @since         3.0
     */
    public IBlob getBlob( String columnName ) throws OdaException;

    /**
     * Gets the value of the designated column in the current row
     * as an IClob object.
     * Note: The driver must guarantee that
     * the returned object and its CLOB data would remain valid
     * and accessible until this result set is closed.
     * @param index   column number (1-based)
     * @return        an IClob object that represents the CLOB value
     *                      in the specific column of the current row;
     *                      or <code>null</code> if the specific
     *                      column has null value
     * @throws OdaException       if data source error occurs
     * @since         3.0
     */
    public IClob getClob( int index ) throws OdaException;

    /**
     * Gets the value of the designated column in the current row
     * as an IClob object.
     * Note: The driver must guarantee that
     * the returned object and its CLOB data would remain valid
     * and accessible until this result set is closed.
     * @param columnName    column name
     * @return        an IClob object that represents the CLOB value
     *                      in the specific column of the current row;
     *                      or <code>null</code> if the specific
     *                      column has null value
     * @throws OdaException       if data source error occurs
     * @since         3.0
     */
    public IClob getClob( String columnName ) throws OdaException;
```

3. Added APIs:

3.1 IBlob

Component name = ODA Run-time Interfaces

Package name = org.eclipse.datatools.connectivity.oda

3.1.1 Change Request: Support of CLOB and BLOB data types.

Proposed Solution:

Add an interface IBlob to encapsulate the processing of a BLOB data value.

```
/**
 * An optional interface that represents a Binary Large Object (BLOB)
value.
 * <br>The interface must be implemented only if the ODA driver
 * supports the BLOB data type.
 * <p>The IBlob interface provides methods for retrieving a BLOB
value
 * as a Java input stream that can be read in smaller chunks, and
 * for optionally getting the length of a BLOB value.
 * <br>
 * The interface method <code>IResultSet.getBlob</code> returns
 * an IBlob instance.
 * @since      3.0
 */
public interface IBlob
{
    /**
     * Retrieves the BLOB value designated by this IBlob instance
     * as a binary stream of uninterpreted bytes.
     * @return a Java input stream that delivers the BLOB data
     *               as a stream of uninterpreted bytes
     * @throws OdaException            if data source error occurs
     */
    public InputStream getBinaryStream() throws OdaException;

    /**
     * Retrieves all or part of the BLOB value designated by this
     * IBlob instance as an array of bytes.
     * <br>An optional short-cut method to retrieve from the
     * instance's binary stream.
     * @param position the 1-based ordinal position of the first byte
     *                 in the BLOB value to be extracted
     * @param length   the number of consecutive bytes to be copied
     * @return a byte array containing up to <code>length</code>
     *         consecutive bytes from the BLOB value,
     *         starting with the byte at <code>position</code>
     * @throws OdaException            if data source error occurs
     */
    public byte[] getBytes( long position, int length )
         throws OdaException;

    /**
     * Returns the number of bytes in the BLOB value designated
     * by this IBlob object.
     * An optional method; throws UnsupportedOperationException
     * if a driver does not support retrieving the length.
     * @return length of the BLOB value in bytes
     * @throws OdaException            if data source error occurs
     */
    public long length() throws OdaException;
}
```

3.2 IClob

Component name = ODA Run-time Interfaces

Package name = org.eclipse.datatools.connectivity.oda

3.2.1 Change Request: Support of CLOB and BLOB data types.

Proposed Solution:

Add an interface IClob to encapsulate the processing of a CLOB data value.

```
/**
 * An optional interface that represents a Character Large Object
(CLOB) value.
 * <br>The interface must be implemented only if the ODA driver
 * supports the CLOB data type.
 * <p>The IClob interface provides methods for retrieving a CLOB
value
 * as a Java stream that can be read in smaller chunks, and
 * for optionally getting the length of a CLOB value.
 * <br>
 * The interface method <code>IResultSet.getClob</code> returns
 * an IClob instance.
 * @since      3.0
 */
public interface IClob
{
    /**
     * Retrieves the CLOB value designated by this IClob instance
     * as a java.io.Reader object for reading a stream of characters.
     * @return a java.io.Reader object that contains the CLOB data
     * @throws OdaException            if data source error occurs
     */
    public Reader getCharacterStream() throws OdaException;

    /**
     * Retrieves a copy of the specified substring in the CLOB value
     * designated by this IClob instance.
     * <br>An optional short-cut method to retrieve from the
     * instance's character stream.
     * @param position the first character of the substring to be
                       extracted.
     *                 The first character is at position 1.
     * @param length   the number of consecutive characters to be
                       copied
     * @return the specified substring that begins at position
     *               and has up to length consecutive characters.
     * @throws OdaException      if data source error occurs
     */
    public String getSubString( long position, int length )
         throws OdaException;

    /**
     * Returns the number of characters in the CLOB value
     * designated by this IClob object.
     * An optional method; throws UnsupportedOperationException
     * if a driver does not support retrieving the length.
     * @return length of the CLOB value in characters
     * @throws OdaException            if data source error occurs
```

```
     */
    public long length() throws OdaException;
}
```

4. Removed APIs:

None.

5. Miscellaneous Change Requests

5.1 datasource.exsd

Component name = ODA Plug-in Extension Point Schema Definition

Package name = org.eclipse.datatools.connectivity.oda.dataSource

Change Request:

Defines the mapping of an ODA data source's native data type to the ODA Blob or Clob data type.

Proposed Solution:

Add ODA scalar data type names for the Blob and Clob data types.

```
<attribute name="odaScalarDataType" use="required" >
    <simpleType>
       <restriction base="string">
          <enumeration value="Date">
          </enumeration>
          <enumeration value="Double">
          </enumeration>
          <enumeration value="Integer">
          </enumeration>
          <enumeration value="String">
          </enumeration>
          <enumeration value="Time">
          </enumeration>
          <enumeration value="Timestamp">
          </enumeration>
          <enumeration value="Decimal">
          </enumeration>
          <enumeration value="Blob">
          </enumeration>
          <enumeration value="Clob">
          </enumeration>
       </restriction>
    </simpleType>
</attribute>
```