| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SyncResolver.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/sql/rowset/spi/SyncProviderException.html)   [**NEXT CLASS**](http://docs.google.com/javax/sql/rowset/spi/TransactionalWriter.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sql/rowset/spi/SyncResolver.html)    [**NO FRAMES**](http://docs.google.com/SyncResolver.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#tyjcwt) | DETAIL: [FIELD](#2s8eyo1) | CONSTR | [METHOD](#35nkun2) |

## **javax.sql.rowset.spi**

Interface SyncResolver

**All Superinterfaces:** [ResultSet](http://docs.google.com/java/sql/ResultSet.html), [RowSet](http://docs.google.com/javax/sql/RowSet.html), [Wrapper](http://docs.google.com/java/sql/Wrapper.html)

public interface **SyncResolver**extends [RowSet](http://docs.google.com/javax/sql/RowSet.html)

Defines a framework that allows applications to use a manual decision tree to decide what should be done when a synchronization conflict occurs. Although it is not mandatory for applications to resolve synchronization conflicts manually, this framework provides the means to delegate to the application when conflicts arise.

Note that a conflict is a situation where the RowSet object's original values for a row do not match the values in the data source, which indicates that the data source row has been modified since the last synchronization. Note also that a RowSet object's original values are the values it had just prior to the the last synchronization, which are not necessarily its initial values.

## Description of a SyncResolver Object

A SyncResolver object is a specialized RowSet object that implements the SyncResolver interface. It **may** operate as either a connected RowSet object (an implementation of the JdbcRowSet interface) or a connected RowSet object (an implementation of the CachedRowSet interface or one of its subinterfaces). For information on the subinterfaces, see the [javax.sql.rowset](http://docs.google.com/package-summary.html) package description. The reference implementation for SyncResolver implements the CachedRowSet interface, but other implementations may choose to implement the JdbcRowSet interface to satisfy particular needs.

After an application has attempted to synchronize a RowSet object with the data source (by calling the CachedRowSet method acceptChanges), and one or more conflicts have been found, a rowset's SyncProvider object creates an instance of SyncResolver. This new SyncResolver object has the same number of rows and columns as the RowSet object that was attempting the synchronization. The SyncResolver object contains the values from the data source that caused the conflict(s) and null for all other values. In addition, it contains information about each conflict.

## Getting and Using a SyncResolver Object

When the method acceptChanges encounters conflicts, the SyncProvider object creates a SyncProviderException object and sets it with the new SyncResolver object. The method acceptChanges will throw this exception, which the application can then catch and use to retrieve the SyncResolver object it contains. The following code snippet uses the SyncProviderException method getSyncResolver to get the SyncResolver object *resolver*.

} catch (SyncProviderException spe) {  
 SyncResolver resolver = spe.getSyncResolver();  
 ...  
 }

With *resolver* in hand, an application can use it to get the information it contains about the conflict or conflicts. A SyncResolver object such as *resolver* keeps track of the conflicts for each row in which there is a conflict. It also places a lock on the table or tables affected by the rowset's command so that no more conflicts can occur while the current conflicts are being resolved.

The following kinds of information can be obtained from a SyncResolver object:

* What operation was being attempted when a conflict occurred  
  The SyncProvider interface defines four constants describing states that may occur. Three constants describe the type of operation (update, delete, or insert) that a RowSet object was attempting to perform when a conflict was discovered, and the fourth indicates that there is no conflict. These constants are the possible return values when a SyncResolver object calls the method getStatus.  
   int operation = resolver.getStatus();

## The value in the data source that caused a conflict A conflict exists when a value that a RowSet object has changed and is attempting to write to the data source has also been changed in the data source since the last synchronization. An application can call the SyncResolver method getConflictValue to retrieve the value in the data source that is the cause of the conflict because the values in a SyncResolver object are the conflict values from the data source. java.lang.Object conflictValue = resolver.getConflictValue(2); Note that the column in *resolver* can be designated by the column number, as is done in the preceding line of code, or by the column name. With the information retrieved from the methods getStatus and getConflictValue, the application may make a determination as to which value should be persisted in the data source. The application then calls the SyncResolver method setResolvedValue, which sets the value to be persisted in the RowSet object and also in the data source. resolver.setResolvedValue("DEPT", 8390426); In the preceding line of code, the column name designates the column in the RowSet object that is to be set with the given value. The column number can also be used to designate the column. An application calls the method setResolvedValue after it has resolved all of the conflicts in the current conflict row and repeats this process for each conflict row in the SyncResolver object. Navigating a SyncResolver ObjectBecause a SyncResolver object is a RowSet object, an application can use all of the RowSet methods for moving the cursor to navigate a SyncResolver object. For example, an application can use the RowSet method next to get to each row and then call the SyncResolver method getStatus to see if the row contains a conflict. In a row with one or more conflicts, the application can iterate through the columns to find any non-null values, which will be the values from the data source that are in conflict. To make it easier to navigate a SyncResolver object, especially when there are large numbers of rows with no conflicts, the SyncResolver interface defines the methods nextConflict and previousConflict, which move only to rows that contain at least one conflict value. Then an application can call the SyncResolver method getConflictValue, supplying it with the column number, to get the conflict value itself. The code fragment in the next section gives an example. Code ExampleThe following code fragment demonstrates how a disconnected RowSet object *crs* might attempt to synchronize itself with the underlying data source and then resolve the conflicts. In the try block, *crs* calls the method acceptChanges, passing it the Connection object *con*. If there are no conflicts, the changes in *crs* are simply written to the data source. However, if there is a conflict, the method acceptChanges throws a SyncProviderException object, and the catch block takes effect. In this example, which illustrates one of the many ways a SyncResolver object can be used, the SyncResolver method nextConflict is used in a while loop. The loop will end when nextConflict returns false, which will occur when there are no more conflict rows in the SyncResolver object *resolver*. In This particular code fragment, *resolver* looks for rows that have update conflicts (rows with the status SyncResolver.UPDATE\_ROW\_CONFLICT), and the rest of this code fragment executes only for rows where conflicts occurred because *crs* was attempting an update. After the cursor for *resolver* has moved to the next conflict row that has an update conflict, the method getRow indicates the number of the current row, and the cursor for the CachedRowSet object *crs* is moved to the comparable row in *crs*. By iterating through the columns of that row in both *resolver* and *crs*, the conflicting values can be retrieved and compared to decide which one should be persisted. In this code fragment, the value in *crs* is the one set as the resolved value, which means that it will be used to overwrite the conflict value in the data source. try { crs.acceptChanges(con); } catch (SyncProviderException spe) { SyncResolver resolver = spe.getSyncResolver(); Object crsValue; // value in the RowSet object Object resolverValue: // value in the SyncResolver object Object resolvedValue: // value to be persisted while(resolver.nextConflict()) { if(resolver.getStatus() == SyncResolver.UPDATE\_ROW\_CONFLICT) { int row = resolver.getRow(); crs.absolute(row); int colCount = crs.getMetaData().getColumnCount(); for(int j = 1; j <= colCount; j++) { if (resolver.getConflictValue(j) != null) { crsValue = crs.getObject(j); resolverValue = resolver.getConflictValue(j); . . . // compare crsValue and resolverValue to determine // which should be the resolved value (the value to persist) resolvedValue = crsValue; resolver.setResolvedValue(j, resolvedValue); } } } } }

| **Field Summary** | |
| --- | --- |
| static int | [**DELETE\_ROW\_CONFLICT**](http://docs.google.com/javax/sql/rowset/spi/SyncResolver.html#DELETE_ROW_CONFLICT)           Indicates that a conflict occurred while the RowSet object was attempting to delete a row in the data source. |
| static int | [**INSERT\_ROW\_CONFLICT**](http://docs.google.com/javax/sql/rowset/spi/SyncResolver.html#INSERT_ROW_CONFLICT)           Indicates that a conflict occurred while the RowSet object was attempting to insert a row into the data source. |
| static int | [**NO\_ROW\_CONFLICT**](http://docs.google.com/javax/sql/rowset/spi/SyncResolver.html#NO_ROW_CONFLICT)           Indicates that **no** conflict occured while the RowSet object was attempting to update, delete or insert a row in the data source. |
| static int | [**UPDATE\_ROW\_CONFLICT**](http://docs.google.com/javax/sql/rowset/spi/SyncResolver.html#UPDATE_ROW_CONFLICT)           Indicates that a conflict occurred while the RowSet object was attempting to update a row in the data source. |

| **Fields inherited from interface java.sql.**[**ResultSet**](http://docs.google.com/java/sql/ResultSet.html) |
| --- |
| [CLOSE\_CURSORS\_AT\_COMMIT](http://docs.google.com/java/sql/ResultSet.html#CLOSE_CURSORS_AT_COMMIT), [CONCUR\_READ\_ONLY](http://docs.google.com/java/sql/ResultSet.html#CONCUR_READ_ONLY), [CONCUR\_UPDATABLE](http://docs.google.com/java/sql/ResultSet.html#CONCUR_UPDATABLE), [FETCH\_FORWARD](http://docs.google.com/java/sql/ResultSet.html#FETCH_FORWARD), [FETCH\_REVERSE](http://docs.google.com/java/sql/ResultSet.html#FETCH_REVERSE), [FETCH\_UNKNOWN](http://docs.google.com/java/sql/ResultSet.html#FETCH_UNKNOWN), [HOLD\_CURSORS\_OVER\_COMMIT](http://docs.google.com/java/sql/ResultSet.html#HOLD_CURSORS_OVER_COMMIT), [TYPE\_FORWARD\_ONLY](http://docs.google.com/java/sql/ResultSet.html#TYPE_FORWARD_ONLY), [TYPE\_SCROLL\_INSENSITIVE](http://docs.google.com/java/sql/ResultSet.html#TYPE_SCROLL_INSENSITIVE), [TYPE\_SCROLL\_SENSITIVE](http://docs.google.com/java/sql/ResultSet.html#TYPE_SCROLL_SENSITIVE) |

| **Method Summary** | |
| --- | --- |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getConflictValue**](http://docs.google.com/javax/sql/rowset/spi/SyncResolver.html#getConflictValue(int))(int index)           Retrieves the value in the designated column in the current row of this SyncResolver object, which is the value in the data source that caused a conflict. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getConflictValue**](http://docs.google.com/javax/sql/rowset/spi/SyncResolver.html#getConflictValue(java.lang.String))([String](http://docs.google.com/java/lang/String.html) columnName)           Retrieves the value in the designated column in the current row of this SyncResolver object, which is the value in the data source that caused a conflict. |
| int | [**getStatus**](http://docs.google.com/javax/sql/rowset/spi/SyncResolver.html#getStatus())()           Retrieves the conflict status of the current row of this SyncResolver, which indicates the operation the RowSet object was attempting when the conflict occurred. |
| boolean | [**nextConflict**](http://docs.google.com/javax/sql/rowset/spi/SyncResolver.html#nextConflict())()           Moves the cursor down from its current position to the next row that contains a conflict value. |
| boolean | [**previousConflict**](http://docs.google.com/javax/sql/rowset/spi/SyncResolver.html#previousConflict())()           Moves the cursor up from its current position to the previous conflict row in this SyncResolver object. |
| void | [**setResolvedValue**](http://docs.google.com/javax/sql/rowset/spi/SyncResolver.html#setResolvedValue(int,%20java.lang.Object))(int index, [Object](http://docs.google.com/java/lang/Object.html) obj)           Sets *obj* as the value in column *index* in the current row of the RowSet object that is being synchronized. |
| void | [**setResolvedValue**](http://docs.google.com/javax/sql/rowset/spi/SyncResolver.html#setResolvedValue(java.lang.String,%20java.lang.Object))([String](http://docs.google.com/java/lang/String.html) columnName, [Object](http://docs.google.com/java/lang/Object.html) obj)           Sets *obj* as the value in column *columnName* in the current row of the RowSet object that is being synchronized. |

| **Methods inherited from interface javax.sql.**[**RowSet**](http://docs.google.com/javax/sql/RowSet.html) |
| --- |
| [addRowSetListener](http://docs.google.com/javax/sql/RowSet.html#addRowSetListener(javax.sql.RowSetListener)), [clearParameters](http://docs.google.com/javax/sql/RowSet.html#clearParameters()), [execute](http://docs.google.com/javax/sql/RowSet.html#execute()), [getCommand](http://docs.google.com/javax/sql/RowSet.html#getCommand()), [getDataSourceName](http://docs.google.com/javax/sql/RowSet.html#getDataSourceName()), [getEscapeProcessing](http://docs.google.com/javax/sql/RowSet.html#getEscapeProcessing()), [getMaxFieldSize](http://docs.google.com/javax/sql/RowSet.html#getMaxFieldSize()), [getMaxRows](http://docs.google.com/javax/sql/RowSet.html#getMaxRows()), [getPassword](http://docs.google.com/javax/sql/RowSet.html#getPassword()), [getQueryTimeout](http://docs.google.com/javax/sql/RowSet.html#getQueryTimeout()), [getTransactionIsolation](http://docs.google.com/javax/sql/RowSet.html#getTransactionIsolation()), [getTypeMap](http://docs.google.com/javax/sql/RowSet.html#getTypeMap()), [getUrl](http://docs.google.com/javax/sql/RowSet.html#getUrl()), [getUsername](http://docs.google.com/javax/sql/RowSet.html#getUsername()), [isReadOnly](http://docs.google.com/javax/sql/RowSet.html#isReadOnly()), [removeRowSetListener](http://docs.google.com/javax/sql/RowSet.html#removeRowSetListener(javax.sql.RowSetListener)), [setArray](http://docs.google.com/javax/sql/RowSet.html#setArray(int,%20java.sql.Array)), [setAsciiStream](http://docs.google.com/javax/sql/RowSet.html#setAsciiStream(int,%20java.io.InputStream)), [setAsciiStream](http://docs.google.com/javax/sql/RowSet.html#setAsciiStream(int,%20java.io.InputStream,%20int)), [setAsciiStream](http://docs.google.com/javax/sql/RowSet.html#setAsciiStream(java.lang.String,%20java.io.InputStream)), [setAsciiStream](http://docs.google.com/javax/sql/RowSet.html#setAsciiStream(java.lang.String,%20java.io.InputStream,%20int)), [setBigDecimal](http://docs.google.com/javax/sql/RowSet.html#setBigDecimal(int,%20java.math.BigDecimal)), [setBigDecimal](http://docs.google.com/javax/sql/RowSet.html#setBigDecimal(java.lang.String,%20java.math.BigDecimal)), [setBinaryStream](http://docs.google.com/javax/sql/RowSet.html#setBinaryStream(int,%20java.io.InputStream)), [setBinaryStream](http://docs.google.com/javax/sql/RowSet.html#setBinaryStream(int,%20java.io.InputStream,%20int)), [setBinaryStream](http://docs.google.com/javax/sql/RowSet.html#setBinaryStream(java.lang.String,%20java.io.InputStream)), [setBinaryStream](http://docs.google.com/javax/sql/RowSet.html#setBinaryStream(java.lang.String,%20java.io.InputStream,%20int)), [setBlob](http://docs.google.com/javax/sql/RowSet.html#setBlob(int,%20java.sql.Blob)), [setBlob](http://docs.google.com/javax/sql/RowSet.html#setBlob(int,%20java.io.InputStream)), [setBlob](http://docs.google.com/javax/sql/RowSet.html#setBlob(int,%20java.io.InputStream,%20long)), [setBlob](http://docs.google.com/javax/sql/RowSet.html#setBlob(java.lang.String,%20java.sql.Blob)), [setBlob](http://docs.google.com/javax/sql/RowSet.html#setBlob(java.lang.String,%20java.io.InputStream)), [setBlob](http://docs.google.com/javax/sql/RowSet.html#setBlob(java.lang.String,%20java.io.InputStream,%20long)), [setBoolean](http://docs.google.com/javax/sql/RowSet.html#setBoolean(int,%20boolean)), [setBoolean](http://docs.google.com/javax/sql/RowSet.html#setBoolean(java.lang.String,%20boolean)), [setByte](http://docs.google.com/javax/sql/RowSet.html#setByte(int,%20byte)), [setByte](http://docs.google.com/javax/sql/RowSet.html#setByte(java.lang.String,%20byte)), [setBytes](http://docs.google.com/javax/sql/RowSet.html#setBytes(int,%20byte%5B%5D)), [setBytes](http://docs.google.com/javax/sql/RowSet.html#setBytes(java.lang.String,%20byte%5B%5D)), [setCharacterStream](http://docs.google.com/javax/sql/RowSet.html#setCharacterStream(int,%20java.io.Reader)), [setCharacterStream](http://docs.google.com/javax/sql/RowSet.html#setCharacterStream(int,%20java.io.Reader,%20int)), [setCharacterStream](http://docs.google.com/javax/sql/RowSet.html#setCharacterStream(java.lang.String,%20java.io.Reader)), [setCharacterStream](http://docs.google.com/javax/sql/RowSet.html#setCharacterStream(java.lang.String,%20java.io.Reader,%20int)), [setClob](http://docs.google.com/javax/sql/RowSet.html#setClob(int,%20java.sql.Clob)), [setClob](http://docs.google.com/javax/sql/RowSet.html#setClob(int,%20java.io.Reader)), [setClob](http://docs.google.com/javax/sql/RowSet.html#setClob(int,%20java.io.Reader,%20long)), [setClob](http://docs.google.com/javax/sql/RowSet.html#setClob(java.lang.String,%20java.sql.Clob)), [setClob](http://docs.google.com/javax/sql/RowSet.html#setClob(java.lang.String,%20java.io.Reader)), [setClob](http://docs.google.com/javax/sql/RowSet.html#setClob(java.lang.String,%20java.io.Reader,%20long)), [setCommand](http://docs.google.com/javax/sql/RowSet.html#setCommand(java.lang.String)), [setConcurrency](http://docs.google.com/javax/sql/RowSet.html#setConcurrency(int)), [setDataSourceName](http://docs.google.com/javax/sql/RowSet.html#setDataSourceName(java.lang.String)), [setDate](http://docs.google.com/javax/sql/RowSet.html#setDate(int,%20java.sql.Date)), [setDate](http://docs.google.com/javax/sql/RowSet.html#setDate(int,%20java.sql.Date,%20java.util.Calendar)), [setDate](http://docs.google.com/javax/sql/RowSet.html#setDate(java.lang.String,%20java.sql.Date)), [setDate](http://docs.google.com/javax/sql/RowSet.html#setDate(java.lang.String,%20java.sql.Date,%20java.util.Calendar)), [setDouble](http://docs.google.com/javax/sql/RowSet.html#setDouble(int,%20double)), [setDouble](http://docs.google.com/javax/sql/RowSet.html#setDouble(java.lang.String,%20double)), [setEscapeProcessing](http://docs.google.com/javax/sql/RowSet.html#setEscapeProcessing(boolean)), [setFloat](http://docs.google.com/javax/sql/RowSet.html#setFloat(int,%20float)), [setFloat](http://docs.google.com/javax/sql/RowSet.html#setFloat(java.lang.String,%20float)), [setInt](http://docs.google.com/javax/sql/RowSet.html#setInt(int,%20int)), [setInt](http://docs.google.com/javax/sql/RowSet.html#setInt(java.lang.String,%20int)), [setLong](http://docs.google.com/javax/sql/RowSet.html#setLong(int,%20long)), [setLong](http://docs.google.com/javax/sql/RowSet.html#setLong(java.lang.String,%20long)), [setMaxFieldSize](http://docs.google.com/javax/sql/RowSet.html#setMaxFieldSize(int)), [setMaxRows](http://docs.google.com/javax/sql/RowSet.html#setMaxRows(int)), [setNCharacterStream](http://docs.google.com/javax/sql/RowSet.html#setNCharacterStream(int,%20java.io.Reader)), [setNCharacterStream](http://docs.google.com/javax/sql/RowSet.html#setNCharacterStream(int,%20java.io.Reader,%20long)), [setNCharacterStream](http://docs.google.com/javax/sql/RowSet.html#setNCharacterStream(java.lang.String,%20java.io.Reader)), [setNCharacterStream](http://docs.google.com/javax/sql/RowSet.html#setNCharacterStream(java.lang.String,%20java.io.Reader,%20long)), [setNClob](http://docs.google.com/javax/sql/RowSet.html#setNClob(int,%20java.sql.NClob)), [setNClob](http://docs.google.com/javax/sql/RowSet.html#setNClob(int,%20java.io.Reader)), [setNClob](http://docs.google.com/javax/sql/RowSet.html#setNClob(int,%20java.io.Reader,%20long)), [setNClob](http://docs.google.com/javax/sql/RowSet.html#setNClob(java.lang.String,%20java.sql.NClob)), [setNClob](http://docs.google.com/javax/sql/RowSet.html#setNClob(java.lang.String,%20java.io.Reader)), [setNClob](http://docs.google.com/javax/sql/RowSet.html#setNClob(java.lang.String,%20java.io.Reader,%20long)), [setNString](http://docs.google.com/javax/sql/RowSet.html#setNString(int,%20java.lang.String)), [setNString](http://docs.google.com/javax/sql/RowSet.html#setNString(java.lang.String,%20java.lang.String)), [setNull](http://docs.google.com/javax/sql/RowSet.html#setNull(int,%20int)), [setNull](http://docs.google.com/javax/sql/RowSet.html#setNull(int,%20int,%20java.lang.String)), [setNull](http://docs.google.com/javax/sql/RowSet.html#setNull(java.lang.String,%20int)), [setNull](http://docs.google.com/javax/sql/RowSet.html#setNull(java.lang.String,%20int,%20java.lang.String)), [setObject](http://docs.google.com/javax/sql/RowSet.html#setObject(int,%20java.lang.Object)), [setObject](http://docs.google.com/javax/sql/RowSet.html#setObject(int,%20java.lang.Object,%20int)), [setObject](http://docs.google.com/javax/sql/RowSet.html#setObject(int,%20java.lang.Object,%20int,%20int)), [setObject](http://docs.google.com/javax/sql/RowSet.html#setObject(java.lang.String,%20java.lang.Object)), [setObject](http://docs.google.com/javax/sql/RowSet.html#setObject(java.lang.String,%20java.lang.Object,%20int)), [setObject](http://docs.google.com/javax/sql/RowSet.html#setObject(java.lang.String,%20java.lang.Object,%20int,%20int)), [setPassword](http://docs.google.com/javax/sql/RowSet.html#setPassword(java.lang.String)), [setQueryTimeout](http://docs.google.com/javax/sql/RowSet.html#setQueryTimeout(int)), [setReadOnly](http://docs.google.com/javax/sql/RowSet.html#setReadOnly(boolean)), [setRef](http://docs.google.com/javax/sql/RowSet.html#setRef(int,%20java.sql.Ref)), [setRowId](http://docs.google.com/javax/sql/RowSet.html#setRowId(int,%20java.sql.RowId)), [setRowId](http://docs.google.com/javax/sql/RowSet.html#setRowId(java.lang.String,%20java.sql.RowId)), [setShort](http://docs.google.com/javax/sql/RowSet.html#setShort(int,%20short)), [setShort](http://docs.google.com/javax/sql/RowSet.html#setShort(java.lang.String,%20short)), [setSQLXML](http://docs.google.com/javax/sql/RowSet.html#setSQLXML(int,%20java.sql.SQLXML)), [setSQLXML](http://docs.google.com/javax/sql/RowSet.html#setSQLXML(java.lang.String,%20java.sql.SQLXML)), [setString](http://docs.google.com/javax/sql/RowSet.html#setString(int,%20java.lang.String)), [setString](http://docs.google.com/javax/sql/RowSet.html#setString(java.lang.String,%20java.lang.String)), [setTime](http://docs.google.com/javax/sql/RowSet.html#setTime(int,%20java.sql.Time)), [setTime](http://docs.google.com/javax/sql/RowSet.html#setTime(int,%20java.sql.Time,%20java.util.Calendar)), [setTime](http://docs.google.com/javax/sql/RowSet.html#setTime(java.lang.String,%20java.sql.Time)), [setTime](http://docs.google.com/javax/sql/RowSet.html#setTime(java.lang.String,%20java.sql.Time,%20java.util.Calendar)), [setTimestamp](http://docs.google.com/javax/sql/RowSet.html#setTimestamp(int,%20java.sql.Timestamp)), [setTimestamp](http://docs.google.com/javax/sql/RowSet.html#setTimestamp(int,%20java.sql.Timestamp,%20java.util.Calendar)), [setTimestamp](http://docs.google.com/javax/sql/RowSet.html#setTimestamp(java.lang.String,%20java.sql.Timestamp)), [setTimestamp](http://docs.google.com/javax/sql/RowSet.html#setTimestamp(java.lang.String,%20java.sql.Timestamp,%20java.util.Calendar)), [setTransactionIsolation](http://docs.google.com/javax/sql/RowSet.html#setTransactionIsolation(int)), [setType](http://docs.google.com/javax/sql/RowSet.html#setType(int)), [setTypeMap](http://docs.google.com/javax/sql/RowSet.html#setTypeMap(java.util.Map)), [setURL](http://docs.google.com/javax/sql/RowSet.html#setURL(int,%20java.net.URL)), [setUrl](http://docs.google.com/javax/sql/RowSet.html#setUrl(java.lang.String)), [setUsername](http://docs.google.com/javax/sql/RowSet.html#setUsername(java.lang.String)) |

| **Methods inherited from interface java.sql.**[**ResultSet**](http://docs.google.com/java/sql/ResultSet.html) |
| --- |
| [absolute](http://docs.google.com/java/sql/ResultSet.html#absolute(int)), [afterLast](http://docs.google.com/java/sql/ResultSet.html#afterLast()), [beforeFirst](http://docs.google.com/java/sql/ResultSet.html#beforeFirst()), [cancelRowUpdates](http://docs.google.com/java/sql/ResultSet.html#cancelRowUpdates()), [clearWarnings](http://docs.google.com/java/sql/ResultSet.html#clearWarnings()), [close](http://docs.google.com/java/sql/ResultSet.html#close()), [deleteRow](http://docs.google.com/java/sql/ResultSet.html#deleteRow()), [findColumn](http://docs.google.com/java/sql/ResultSet.html#findColumn(java.lang.String)), [first](http://docs.google.com/java/sql/ResultSet.html#first()), [getArray](http://docs.google.com/java/sql/ResultSet.html#getArray(int)), [getArray](http://docs.google.com/java/sql/ResultSet.html#getArray(java.lang.String)), [getAsciiStream](http://docs.google.com/java/sql/ResultSet.html#getAsciiStream(int)), [getAsciiStream](http://docs.google.com/java/sql/ResultSet.html#getAsciiStream(java.lang.String)), [getBigDecimal](http://docs.google.com/java/sql/ResultSet.html#getBigDecimal(int)), [getBigDecimal](http://docs.google.com/java/sql/ResultSet.html#getBigDecimal(int,%20int)), [getBigDecimal](http://docs.google.com/java/sql/ResultSet.html#getBigDecimal(java.lang.String)), [getBigDecimal](http://docs.google.com/java/sql/ResultSet.html#getBigDecimal(java.lang.String,%20int)), [getBinaryStream](http://docs.google.com/java/sql/ResultSet.html#getBinaryStream(int)), [getBinaryStream](http://docs.google.com/java/sql/ResultSet.html#getBinaryStream(java.lang.String)), [getBlob](http://docs.google.com/java/sql/ResultSet.html#getBlob(int)), [getBlob](http://docs.google.com/java/sql/ResultSet.html#getBlob(java.lang.String)), [getBoolean](http://docs.google.com/java/sql/ResultSet.html#getBoolean(int)), [getBoolean](http://docs.google.com/java/sql/ResultSet.html#getBoolean(java.lang.String)), [getByte](http://docs.google.com/java/sql/ResultSet.html#getByte(int)), [getByte](http://docs.google.com/java/sql/ResultSet.html#getByte(java.lang.String)), [getBytes](http://docs.google.com/java/sql/ResultSet.html#getBytes(int)), [getBytes](http://docs.google.com/java/sql/ResultSet.html#getBytes(java.lang.String)), [getCharacterStream](http://docs.google.com/java/sql/ResultSet.html#getCharacterStream(int)), [getCharacterStream](http://docs.google.com/java/sql/ResultSet.html#getCharacterStream(java.lang.String)), [getClob](http://docs.google.com/java/sql/ResultSet.html#getClob(int)), [getClob](http://docs.google.com/java/sql/ResultSet.html#getClob(java.lang.String)), [getConcurrency](http://docs.google.com/java/sql/ResultSet.html#getConcurrency()), [getCursorName](http://docs.google.com/java/sql/ResultSet.html#getCursorName()), [getDate](http://docs.google.com/java/sql/ResultSet.html#getDate(int)), [getDate](http://docs.google.com/java/sql/ResultSet.html#getDate(int,%20java.util.Calendar)), [getDate](http://docs.google.com/java/sql/ResultSet.html#getDate(java.lang.String)), [getDate](http://docs.google.com/java/sql/ResultSet.html#getDate(java.lang.String,%20java.util.Calendar)), [getDouble](http://docs.google.com/java/sql/ResultSet.html#getDouble(int)), [getDouble](http://docs.google.com/java/sql/ResultSet.html#getDouble(java.lang.String)), [getFetchDirection](http://docs.google.com/java/sql/ResultSet.html#getFetchDirection()), [getFetchSize](http://docs.google.com/java/sql/ResultSet.html#getFetchSize()), [getFloat](http://docs.google.com/java/sql/ResultSet.html#getFloat(int)), [getFloat](http://docs.google.com/java/sql/ResultSet.html#getFloat(java.lang.String)), [getHoldability](http://docs.google.com/java/sql/ResultSet.html#getHoldability()), [getInt](http://docs.google.com/java/sql/ResultSet.html#getInt(int)), [getInt](http://docs.google.com/java/sql/ResultSet.html#getInt(java.lang.String)), [getLong](http://docs.google.com/java/sql/ResultSet.html#getLong(int)), [getLong](http://docs.google.com/java/sql/ResultSet.html#getLong(java.lang.String)), [getMetaData](http://docs.google.com/java/sql/ResultSet.html#getMetaData()), [getNCharacterStream](http://docs.google.com/java/sql/ResultSet.html#getNCharacterStream(int)), [getNCharacterStream](http://docs.google.com/java/sql/ResultSet.html#getNCharacterStream(java.lang.String)), [getNClob](http://docs.google.com/java/sql/ResultSet.html#getNClob(int)), [getNClob](http://docs.google.com/java/sql/ResultSet.html#getNClob(java.lang.String)), [getNString](http://docs.google.com/java/sql/ResultSet.html#getNString(int)), [getNString](http://docs.google.com/java/sql/ResultSet.html#getNString(java.lang.String)), [getObject](http://docs.google.com/java/sql/ResultSet.html#getObject(int)), [getObject](http://docs.google.com/java/sql/ResultSet.html#getObject(int,%20java.util.Map)), [getObject](http://docs.google.com/java/sql/ResultSet.html#getObject(java.lang.String)), [getObject](http://docs.google.com/java/sql/ResultSet.html#getObject(java.lang.String,%20java.util.Map)), [getRef](http://docs.google.com/java/sql/ResultSet.html#getRef(int)), [getRef](http://docs.google.com/java/sql/ResultSet.html#getRef(java.lang.String)), [getRow](http://docs.google.com/java/sql/ResultSet.html#getRow()), [getRowId](http://docs.google.com/java/sql/ResultSet.html#getRowId(int)), [getRowId](http://docs.google.com/java/sql/ResultSet.html#getRowId(java.lang.String)), [getShort](http://docs.google.com/java/sql/ResultSet.html#getShort(int)), [getShort](http://docs.google.com/java/sql/ResultSet.html#getShort(java.lang.String)), [getSQLXML](http://docs.google.com/java/sql/ResultSet.html#getSQLXML(int)), [getSQLXML](http://docs.google.com/java/sql/ResultSet.html#getSQLXML(java.lang.String)), [getStatement](http://docs.google.com/java/sql/ResultSet.html#getStatement()), [getString](http://docs.google.com/java/sql/ResultSet.html#getString(int)), [getString](http://docs.google.com/java/sql/ResultSet.html#getString(java.lang.String)), [getTime](http://docs.google.com/java/sql/ResultSet.html#getTime(int)), [getTime](http://docs.google.com/java/sql/ResultSet.html#getTime(int,%20java.util.Calendar)), [getTime](http://docs.google.com/java/sql/ResultSet.html#getTime(java.lang.String)), [getTime](http://docs.google.com/java/sql/ResultSet.html#getTime(java.lang.String,%20java.util.Calendar)), [getTimestamp](http://docs.google.com/java/sql/ResultSet.html#getTimestamp(int)), [getTimestamp](http://docs.google.com/java/sql/ResultSet.html#getTimestamp(int,%20java.util.Calendar)), [getTimestamp](http://docs.google.com/java/sql/ResultSet.html#getTimestamp(java.lang.String)), [getTimestamp](http://docs.google.com/java/sql/ResultSet.html#getTimestamp(java.lang.String,%20java.util.Calendar)), [getType](http://docs.google.com/java/sql/ResultSet.html#getType()), [getUnicodeStream](http://docs.google.com/java/sql/ResultSet.html#getUnicodeStream(int)), [getUnicodeStream](http://docs.google.com/java/sql/ResultSet.html#getUnicodeStream(java.lang.String)), [getURL](http://docs.google.com/java/sql/ResultSet.html#getURL(int)), [getURL](http://docs.google.com/java/sql/ResultSet.html#getURL(java.lang.String)), [getWarnings](http://docs.google.com/java/sql/ResultSet.html#getWarnings()), [insertRow](http://docs.google.com/java/sql/ResultSet.html#insertRow()), [isAfterLast](http://docs.google.com/java/sql/ResultSet.html#isAfterLast()), [isBeforeFirst](http://docs.google.com/java/sql/ResultSet.html#isBeforeFirst()), [isClosed](http://docs.google.com/java/sql/ResultSet.html#isClosed()), [isFirst](http://docs.google.com/java/sql/ResultSet.html#isFirst()), [isLast](http://docs.google.com/java/sql/ResultSet.html#isLast()), [last](http://docs.google.com/java/sql/ResultSet.html#last()), [moveToCurrentRow](http://docs.google.com/java/sql/ResultSet.html#moveToCurrentRow()), [moveToInsertRow](http://docs.google.com/java/sql/ResultSet.html#moveToInsertRow()), [next](http://docs.google.com/java/sql/ResultSet.html#next()), [previous](http://docs.google.com/java/sql/ResultSet.html#previous()), [refreshRow](http://docs.google.com/java/sql/ResultSet.html#refreshRow()), [relative](http://docs.google.com/java/sql/ResultSet.html#relative(int)), [rowDeleted](http://docs.google.com/java/sql/ResultSet.html#rowDeleted()), [rowInserted](http://docs.google.com/java/sql/ResultSet.html#rowInserted()), [rowUpdated](http://docs.google.com/java/sql/ResultSet.html#rowUpdated()), [setFetchDirection](http://docs.google.com/java/sql/ResultSet.html#setFetchDirection(int)), [setFetchSize](http://docs.google.com/java/sql/ResultSet.html#setFetchSize(int)), [updateArray](http://docs.google.com/java/sql/ResultSet.html#updateArray(int,%20java.sql.Array)), [updateArray](http://docs.google.com/java/sql/ResultSet.html#updateArray(java.lang.String,%20java.sql.Array)), [updateAsciiStream](http://docs.google.com/java/sql/ResultSet.html#updateAsciiStream(int,%20java.io.InputStream)), [updateAsciiStream](http://docs.google.com/java/sql/ResultSet.html#updateAsciiStream(int,%20java.io.InputStream,%20int)), [updateAsciiStream](http://docs.google.com/java/sql/ResultSet.html#updateAsciiStream(int,%20java.io.InputStream,%20long)), [updateAsciiStream](http://docs.google.com/java/sql/ResultSet.html#updateAsciiStream(java.lang.String,%20java.io.InputStream)), [updateAsciiStream](http://docs.google.com/java/sql/ResultSet.html#updateAsciiStream(java.lang.String,%20java.io.InputStream,%20int)), [updateAsciiStream](http://docs.google.com/java/sql/ResultSet.html#updateAsciiStream(java.lang.String,%20java.io.InputStream,%20long)), [updateBigDecimal](http://docs.google.com/java/sql/ResultSet.html#updateBigDecimal(int,%20java.math.BigDecimal)), [updateBigDecimal](http://docs.google.com/java/sql/ResultSet.html#updateBigDecimal(java.lang.String,%20java.math.BigDecimal)), [updateBinaryStream](http://docs.google.com/java/sql/ResultSet.html#updateBinaryStream(int,%20java.io.InputStream)), [updateBinaryStream](http://docs.google.com/java/sql/ResultSet.html#updateBinaryStream(int,%20java.io.InputStream,%20int)), [updateBinaryStream](http://docs.google.com/java/sql/ResultSet.html#updateBinaryStream(int,%20java.io.InputStream,%20long)), [updateBinaryStream](http://docs.google.com/java/sql/ResultSet.html#updateBinaryStream(java.lang.String,%20java.io.InputStream)), [updateBinaryStream](http://docs.google.com/java/sql/ResultSet.html#updateBinaryStream(java.lang.String,%20java.io.InputStream,%20int)), [updateBinaryStream](http://docs.google.com/java/sql/ResultSet.html#updateBinaryStream(java.lang.String,%20java.io.InputStream,%20long)), [updateBlob](http://docs.google.com/java/sql/ResultSet.html#updateBlob(int,%20java.sql.Blob)), [updateBlob](http://docs.google.com/java/sql/ResultSet.html#updateBlob(int,%20java.io.InputStream)), [updateBlob](http://docs.google.com/java/sql/ResultSet.html#updateBlob(int,%20java.io.InputStream,%20long)), [updateBlob](http://docs.google.com/java/sql/ResultSet.html#updateBlob(java.lang.String,%20java.sql.Blob)), [updateBlob](http://docs.google.com/java/sql/ResultSet.html#updateBlob(java.lang.String,%20java.io.InputStream)), [updateBlob](http://docs.google.com/java/sql/ResultSet.html#updateBlob(java.lang.String,%20java.io.InputStream,%20long)), [updateBoolean](http://docs.google.com/java/sql/ResultSet.html#updateBoolean(int,%20boolean)), [updateBoolean](http://docs.google.com/java/sql/ResultSet.html#updateBoolean(java.lang.String,%20boolean)), [updateByte](http://docs.google.com/java/sql/ResultSet.html#updateByte(int,%20byte)), [updateByte](http://docs.google.com/java/sql/ResultSet.html#updateByte(java.lang.String,%20byte)), [updateBytes](http://docs.google.com/java/sql/ResultSet.html#updateBytes(int,%20byte%5B%5D)), [updateBytes](http://docs.google.com/java/sql/ResultSet.html#updateBytes(java.lang.String,%20byte%5B%5D)), [updateCharacterStream](http://docs.google.com/java/sql/ResultSet.html#updateCharacterStream(int,%20java.io.Reader)), [updateCharacterStream](http://docs.google.com/java/sql/ResultSet.html#updateCharacterStream(int,%20java.io.Reader,%20int)), [updateCharacterStream](http://docs.google.com/java/sql/ResultSet.html#updateCharacterStream(int,%20java.io.Reader,%20long)), [updateCharacterStream](http://docs.google.com/java/sql/ResultSet.html#updateCharacterStream(java.lang.String,%20java.io.Reader)), [updateCharacterStream](http://docs.google.com/java/sql/ResultSet.html#updateCharacterStream(java.lang.String,%20java.io.Reader,%20int)), [updateCharacterStream](http://docs.google.com/java/sql/ResultSet.html#updateCharacterStream(java.lang.String,%20java.io.Reader,%20long)), [updateClob](http://docs.google.com/java/sql/ResultSet.html#updateClob(int,%20java.sql.Clob)), [updateClob](http://docs.google.com/java/sql/ResultSet.html#updateClob(int,%20java.io.Reader)), [updateClob](http://docs.google.com/java/sql/ResultSet.html#updateClob(int,%20java.io.Reader,%20long)), [updateClob](http://docs.google.com/java/sql/ResultSet.html#updateClob(java.lang.String,%20java.sql.Clob)), [updateClob](http://docs.google.com/java/sql/ResultSet.html#updateClob(java.lang.String,%20java.io.Reader)), [updateClob](http://docs.google.com/java/sql/ResultSet.html#updateClob(java.lang.String,%20java.io.Reader,%20long)), [updateDate](http://docs.google.com/java/sql/ResultSet.html#updateDate(int,%20java.sql.Date)), [updateDate](http://docs.google.com/java/sql/ResultSet.html#updateDate(java.lang.String,%20java.sql.Date)), [updateDouble](http://docs.google.com/java/sql/ResultSet.html#updateDouble(int,%20double)), [updateDouble](http://docs.google.com/java/sql/ResultSet.html#updateDouble(java.lang.String,%20double)), [updateFloat](http://docs.google.com/java/sql/ResultSet.html#updateFloat(int,%20float)), [updateFloat](http://docs.google.com/java/sql/ResultSet.html#updateFloat(java.lang.String,%20float)), [updateInt](http://docs.google.com/java/sql/ResultSet.html#updateInt(int,%20int)), [updateInt](http://docs.google.com/java/sql/ResultSet.html#updateInt(java.lang.String,%20int)), [updateLong](http://docs.google.com/java/sql/ResultSet.html#updateLong(int,%20long)), [updateLong](http://docs.google.com/java/sql/ResultSet.html#updateLong(java.lang.String,%20long)), [updateNCharacterStream](http://docs.google.com/java/sql/ResultSet.html#updateNCharacterStream(int,%20java.io.Reader)), [updateNCharacterStream](http://docs.google.com/java/sql/ResultSet.html#updateNCharacterStream(int,%20java.io.Reader,%20long)), [updateNCharacterStream](http://docs.google.com/java/sql/ResultSet.html#updateNCharacterStream(java.lang.String,%20java.io.Reader)), [updateNCharacterStream](http://docs.google.com/java/sql/ResultSet.html#updateNCharacterStream(java.lang.String,%20java.io.Reader,%20long)), [updateNClob](http://docs.google.com/java/sql/ResultSet.html#updateNClob(int,%20java.sql.NClob)), [updateNClob](http://docs.google.com/java/sql/ResultSet.html#updateNClob(int,%20java.io.Reader)), [updateNClob](http://docs.google.com/java/sql/ResultSet.html#updateNClob(int,%20java.io.Reader,%20long)), [updateNClob](http://docs.google.com/java/sql/ResultSet.html#updateNClob(java.lang.String,%20java.sql.NClob)), [updateNClob](http://docs.google.com/java/sql/ResultSet.html#updateNClob(java.lang.String,%20java.io.Reader)), [updateNClob](http://docs.google.com/java/sql/ResultSet.html#updateNClob(java.lang.String,%20java.io.Reader,%20long)), [updateNString](http://docs.google.com/java/sql/ResultSet.html#updateNString(int,%20java.lang.String)), [updateNString](http://docs.google.com/java/sql/ResultSet.html#updateNString(java.lang.String,%20java.lang.String)), [updateNull](http://docs.google.com/java/sql/ResultSet.html#updateNull(int)), [updateNull](http://docs.google.com/java/sql/ResultSet.html#updateNull(java.lang.String)), [updateObject](http://docs.google.com/java/sql/ResultSet.html#updateObject(int,%20java.lang.Object)), [updateObject](http://docs.google.com/java/sql/ResultSet.html#updateObject(int,%20java.lang.Object,%20int)), [updateObject](http://docs.google.com/java/sql/ResultSet.html#updateObject(java.lang.String,%20java.lang.Object)), [updateObject](http://docs.google.com/java/sql/ResultSet.html#updateObject(java.lang.String,%20java.lang.Object,%20int)), [updateRef](http://docs.google.com/java/sql/ResultSet.html#updateRef(int,%20java.sql.Ref)), [updateRef](http://docs.google.com/java/sql/ResultSet.html#updateRef(java.lang.String,%20java.sql.Ref)), [updateRow](http://docs.google.com/java/sql/ResultSet.html#updateRow()), [updateRowId](http://docs.google.com/java/sql/ResultSet.html#updateRowId(int,%20java.sql.RowId)), [updateRowId](http://docs.google.com/java/sql/ResultSet.html#updateRowId(java.lang.String,%20java.sql.RowId)), [updateShort](http://docs.google.com/java/sql/ResultSet.html#updateShort(int,%20short)), [updateShort](http://docs.google.com/java/sql/ResultSet.html#updateShort(java.lang.String,%20short)), [updateSQLXML](http://docs.google.com/java/sql/ResultSet.html#updateSQLXML(int,%20java.sql.SQLXML)), [updateSQLXML](http://docs.google.com/java/sql/ResultSet.html#updateSQLXML(java.lang.String,%20java.sql.SQLXML)), [updateString](http://docs.google.com/java/sql/ResultSet.html#updateString(int,%20java.lang.String)), [updateString](http://docs.google.com/java/sql/ResultSet.html#updateString(java.lang.String,%20java.lang.String)), [updateTime](http://docs.google.com/java/sql/ResultSet.html#updateTime(int,%20java.sql.Time)), [updateTime](http://docs.google.com/java/sql/ResultSet.html#updateTime(java.lang.String,%20java.sql.Time)), [updateTimestamp](http://docs.google.com/java/sql/ResultSet.html#updateTimestamp(int,%20java.sql.Timestamp)), [updateTimestamp](http://docs.google.com/java/sql/ResultSet.html#updateTimestamp(java.lang.String,%20java.sql.Timestamp)), [wasNull](http://docs.google.com/java/sql/ResultSet.html#wasNull()) |

| **Methods inherited from interface java.sql.**[**Wrapper**](http://docs.google.com/java/sql/Wrapper.html) |
| --- |
| [isWrapperFor](http://docs.google.com/java/sql/Wrapper.html#isWrapperFor(java.lang.Class)), [unwrap](http://docs.google.com/java/sql/Wrapper.html#unwrap(java.lang.Class)) |

| **Field Detail** |
| --- |

### UPDATE\_ROW\_CONFLICT static final int **UPDATE\_ROW\_CONFLICT**Indicates that a conflict occurred while the RowSet object was attempting to update a row in the data source. The values in the data source row to be updated differ from the RowSet object's original values for that row, which means that the row in the data source has been updated or deleted since the last synchronization. **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.sql.rowset.spi.SyncResolver.UPDATE_ROW_CONFLICT) DELETE\_ROW\_CONFLICT static final int **DELETE\_ROW\_CONFLICT**Indicates that a conflict occurred while the RowSet object was attempting to delete a row in the data source. The values in the data source row to be updated differ from the RowSet object's original values for that row, which means that the row in the data source has been updated or deleted since the last synchronization. **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.sql.rowset.spi.SyncResolver.DELETE_ROW_CONFLICT) INSERT\_ROW\_CONFLICT static final int **INSERT\_ROW\_CONFLICT**Indicates that a conflict occurred while the RowSet object was attempting to insert a row into the data source. This means that a row with the same primary key as the row to be inserted has been inserted into the data source since the last synchronization. **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.sql.rowset.spi.SyncResolver.INSERT_ROW_CONFLICT) NO\_ROW\_CONFLICT static final int **NO\_ROW\_CONFLICT**Indicates that **no** conflict occured while the RowSet object was attempting to update, delete or insert a row in the data source. The values in the SyncResolver will contain null values only as an indication that no information in pertitent to the conflict resolution in this row. **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.sql.rowset.spi.SyncResolver.NO_ROW_CONFLICT)

| **Method Detail** |
| --- |

### getStatus int **getStatus**()Retrieves the conflict status of the current row of this SyncResolver, which indicates the operation the RowSet object was attempting when the conflict occurred. **Returns:**one of the following constants: SyncResolver.UPDATE\_ROW\_CONFLICT, SyncResolver.DELETE\_ROW\_CONFLICT, SyncResolver.INSERT\_ROW\_CONFLICT, or SyncResolver.NO\_ROW\_CONFLICT getConflictValue [Object](http://docs.google.com/java/lang/Object.html) **getConflictValue**(int index) throws [SQLException](http://docs.google.com/java/sql/SQLException.html)Retrieves the value in the designated column in the current row of this SyncResolver object, which is the value in the data source that caused a conflict. **Parameters:**index - an int designating the column in this row of this SyncResolver object from which to retrieve the value causing a conflict **Returns:**the value of the designated column in the current row of this SyncResolver object **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs getConflictValue [Object](http://docs.google.com/java/lang/Object.html) **getConflictValue**([String](http://docs.google.com/java/lang/String.html) columnName) throws [SQLException](http://docs.google.com/java/sql/SQLException.html)Retrieves the value in the designated column in the current row of this SyncResolver object, which is the value in the data source that caused a conflict. **Parameters:**columnName - a String object designating the column in this row of this SyncResolver object from which to retrieve the value causing a conflict **Returns:**the value of the designated column in the current row of this SyncResolver object **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs setResolvedValue void **setResolvedValue**(int index, [Object](http://docs.google.com/java/lang/Object.html) obj) throws [SQLException](http://docs.google.com/java/sql/SQLException.html)Sets *obj* as the value in column *index* in the current row of the RowSet object that is being synchronized. *obj* is set as the value in the data source internally. **Parameters:**index - an int giving the number of the column into which to set the value to be persistedobj - an Object that is the value to be set in the RowSet object and persisted in the data source **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs setResolvedValue void **setResolvedValue**([String](http://docs.google.com/java/lang/String.html) columnName, [Object](http://docs.google.com/java/lang/Object.html) obj) throws [SQLException](http://docs.google.com/java/sql/SQLException.html)Sets *obj* as the value in column *columnName* in the current row of the RowSet object that is being synchronized. *obj* is set as the value in the data source internally. **Parameters:**columnName - a String object giving the name of the column into which to set the value to be persistedobj - an Object that is the value to be set in the RowSet object and persisted in the data source **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs nextConflict boolean **nextConflict**() throws [SQLException](http://docs.google.com/java/sql/SQLException.html)Moves the cursor down from its current position to the next row that contains a conflict value. A SyncResolver object's cursor is initially positioned before the first conflict row; the first call to the method nextConflict makes the first conflict row the current row; the second call makes the second conflict row the current row, and so on. A call to the method nextConflict will implicitly close an input stream if one is open and will clear the SyncResolver object's warning chain. **Returns:**true if the new current row is valid; false if there are no more rows **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or the result set type is TYPE\_FORWARD\_ONLY previousConflict boolean **previousConflict**() throws [SQLException](http://docs.google.com/java/sql/SQLException.html)Moves the cursor up from its current position to the previous conflict row in this SyncResolver object. A call to the method previousConflict will implicitly close an input stream if one is open and will clear the SyncResolver object's warning chain. **Returns:**true if the cursor is on a valid row; false if it is off the result set **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or the result set type is TYPE\_FORWARD\_ONLY

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SyncResolver.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform*** ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/sql/rowset/spi/SyncProviderException.html)   [**NEXT CLASS**](http://docs.google.com/javax/sql/rowset/spi/TransactionalWriter.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sql/rowset/spi/SyncResolver.html)    [**NO FRAMES**](http://docs.google.com/SyncResolver.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#tyjcwt) | DETAIL: [FIELD](#2s8eyo1) | CONSTR | [METHOD](#35nkun2) |

* [Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)  
  For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.  
  Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).