| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SyncProvider.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/SyncFactoryException.html)   [**NEXT CLASS**](http://docs.google.com/javax/sql/rowset/spi/SyncProviderException.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sql/rowset/spi/SyncProvider.html)    [**NO FRAMES**](http://docs.google.com/SyncProvider.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#3j2qqm3) | [METHOD](#4i7ojhp) |

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

Class SyncProvider

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **javax.sql.rowset.spi.SyncProvider**

public abstract class **SyncProvider**extends [Object](http://docs.google.com/java/lang/Object.html)

The synchronization mechanism that provides reader/writer capabilities for disconnected RowSet objects. A SyncProvider implementation is a class that extends the SyncProvider abstract class.

A SyncProvider implementation is identified by a unique ID, which is its fully qualified class name. This name must be registered with the SyncFactory SPI, thus making the implementation available to all RowSet implementations. The factory mechanism in the reference implementation uses this name to instantiate the implementation, which can then provide a RowSet object with its reader (a javax.sql.RowSetReader object) and its writer (a javax.sql.RowSetWriter object).

The Jdbc RowSet Implementations specification provides two reference implementations of the SyncProvider abstract class: RIOptimisticProvider and RIXMLProvider. The RIOptimisticProvider can set any RowSet implementation with a RowSetReader object and a RowSetWriter object. However, only the RIXMLProvider implementation can set an XmlReader object and an XmlWriter object. A WebRowSet object uses the XmlReader object to read data in XML format to populate itself with that data. It uses the XmlWriter object to write itself to a stream or java.io.Writer object in XML format.

### 1.0 Naming Convention for Implementations

As a guide to naming SyncProvider implementations, the following should be noted:

* The name for a SyncProvider implementation is its fully qualified class name.
* It is recommended that vendors supply a SyncProvider implementation in a package named providers.

For instance, if a vendor named Fred, Inc. offered a SyncProvider implementation, you could have the following:

Vendor name: Fred, Inc.   
 Domain name of vendor: com.fred  
 Package name: com.fred.providers  
 SyncProvider implementation class name: HighAvailabilityProvider  
  
 Fully qualified class name of SyncProvider implementation:  
 com.fred.providers.HighAvailabilityProvider

The following line of code uses the fully qualified name to register this implementation with the SyncFactory static instance.

SyncFactory.registerProvider(  
 "com.fred.providers.HighAvailabilityProvider");

The default SyncProvider object provided with the reference implementation uses the following name:

com.sun.rowset.providers.RIOptimisticProvider

A vendor can register a SyncProvider implementation class name with Sun Microsystems, Inc. by sending email to jdbc@sun.com. Sun will maintain a database listing the available SyncProvider implementations for use with compliant RowSet implementations. This database will be similar to the one already maintained to list available JDBC drivers.

Vendors should refer to the reference implementation synchronization providers for additional guidance on how to implement a new SyncProvider implementation.

### 2.0 How a RowSet Object Gets Its Provider

A disconnected Rowset object may get access to a SyncProvider object in one of the following two ways:

* Using a constructor  
    
   CachedRowSet crs = new CachedRowSet(  
   "com.fred.providers.HighAvailabilitySyncProvider");
* Using the setSyncProvider method  
   CachedRowSet crs = new CachedRowSet();   
   crs.setSyncProvider("com.fred.providers.HighAvailabilitySyncProvider");

By default, the reference implementations of the RowSet synchronization providers are always available to the Java platform. If no other pluggable synchronization providers have been correctly registered, the SyncFactory will automatically generate an instance of the default SyncProvider reference implementation. Thus, in the preceding code fragment, if no implementation named com.fred.providers.HighAvailabilitySyncProvider has been registered with the SyncFactory instance, *crs* will be assigned the default provider in the reference implementation, which is com.sun.rowset.providers.RIOptimisticProvider.

### 3.0 Violations and Synchronization Issues

If an update between a disconnected RowSet object and a data source violates the original query or the underlying data source constraints, this will result in undefined behavior for all disconnected RowSet implementations and their designated SyncProvider implementations. Not defining the behavior when such violations occur offers greater flexibility for a SyncProvider implementation to determine its own best course of action.

A SyncProvider implementation may choose to implement a specific handler to handle a subset of query violations. However if an original query violation or a more general data source constraint violation is not handled by the SyncProvider implementation, all SyncProvider objects must throw a SyncProviderException.

### 4.0 Updatable SQL VIEWs

It is possible for any disconnected or connected RowSet object to be populated from an SQL query that is formulated originally from an SQL VIEW. While in many cases it is possible for an update to be performed to an underlying view, such an update requires additional metadata, which may vary. The SyncProvider class provides two constants to indicate whether an implementation supports updating an SQL VIEW.

* **NONUPDATABLE\_VIEW\_SYNC** - Indicates that a SyncProvider implementation does not support synchronization with an SQL VIEW as the underlying source of data for the RowSet object.
* **UPDATABLE\_VIEW\_SYNC** - Indicates that a SyncProvider implementation supports synchronization with an SQL VIEW as the underlying source of data.

The default is for a RowSet object not to be updatable if it was populated with data from an SQL VIEW.

### 5.0 SyncProvider Constants

The SyncProvider class provides three sets of constants that are used as return values or parameters for SyncProvider methods. SyncProvider objects may be implemented to perform synchronization between a RowSet object and its underlying data source with varying degrees of of care. The first group of constants indicate how synchronization is handled. For example, GRADE\_NONE indicates that a SyncProvider object will not take any care to see what data is valid and will simply write the RowSet data to the data source. GRADE\_MODIFIED\_AT\_COMMIT indicates that the provider will check only modified data for validity. Other grades check all data for validity or set locks when data is modified or loaded.

1. Constants to indicate the synchronization grade of a SyncProvider object
   * SyncProvider.GRADE\_NONE
   * SyncProvider.GRADE\_MODIFIED\_AT\_COMMIT
   * SyncProvider.GRADE\_CHECK\_ALL\_AT\_COMMIT
   * SyncProvider.GRADE\_LOCK\_WHEN\_MODIFIED
   * SyncProvider.GRADE\_LOCK\_WHEN\_LOADED
2. Constants to indicate what locks are set on the data source
   * SyncProvider.DATASOURCE\_NO\_LOCK
   * SyncProvider.DATASOURCE\_ROW\_LOCK
   * SyncProvider.DATASOURCE\_TABLE\_LOCK
   * SyncProvider.DATASOURCE\_DB\_LOCK
3. Constants to indicate whether a SyncProvider object can perform updates to an SQL VIEW  
   These constants are explained in the preceding section (4.0).
   * SyncProvider.UPDATABLE\_VIEW\_SYNC
   * SyncProvider.NONUPDATABLE\_VIEW\_SYNC

**See Also:**[SyncFactory](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html), [SyncFactoryException](http://docs.google.com/javax/sql/rowset/spi/SyncFactoryException.html)

| **Field Summary** | |
| --- | --- |
| static int | [**DATASOURCE\_DB\_LOCK**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#DATASOURCE_DB_LOCK)            Indicates that a lock is placed on the entire data source that is the source of data for the RowSet object that is using this SyncProvider object. |
| static int | [**DATASOURCE\_NO\_LOCK**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#DATASOURCE_NO_LOCK)            Indicates that no locks remain on the originating data source. |
| static int | [**DATASOURCE\_ROW\_LOCK**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#DATASOURCE_ROW_LOCK)            Indicates that a lock is placed on the rows that are touched by the original SQL statement used to populate the RowSet object that is using this SyncProvider object. |
| static int | [**DATASOURCE\_TABLE\_LOCK**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#DATASOURCE_TABLE_LOCK)            Indicates that a lock is placed on all tables that are touched by the original SQL statement used to populate the RowSet object that is using this SyncProvider object. |
| static int | [**GRADE\_CHECK\_ALL\_AT\_COMMIT**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#GRADE_CHECK_ALL_AT_COMMIT)            Indicates a high level optimistic synchronization grade with respect to the originating data source. |
| static int | [**GRADE\_CHECK\_MODIFIED\_AT\_COMMIT**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#GRADE_CHECK_MODIFIED_AT_COMMIT)            Indicates a low level optimistic synchronization grade with respect to the originating data source. |
| static int | [**GRADE\_LOCK\_WHEN\_LOADED**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#GRADE_LOCK_WHEN_LOADED)            Indicates the most pessimistic synchronization grade with respect to the originating data source. |
| static int | [**GRADE\_LOCK\_WHEN\_MODIFIED**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#GRADE_LOCK_WHEN_MODIFIED)            Indicates a pessimistic synchronization grade with respect to the originating data source. |
| static int | [**GRADE\_NONE**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#GRADE_NONE)            Indicates that no synchronization with the originating data source is provided. |
| static int | [**NONUPDATABLE\_VIEW\_SYNC**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#NONUPDATABLE_VIEW_SYNC)            Indicates that a SyncProvider implementation does **not** support synchronization between a RowSet object and the SQL VIEW used to populate it. |
| static int | [**UPDATABLE\_VIEW\_SYNC**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#UPDATABLE_VIEW_SYNC)            Indicates that a SyncProvider implementation supports synchronization between a RowSet object and the SQL VIEW used to populate it. |

| **Constructor Summary** | |
| --- | --- |
| [**SyncProvider**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#SyncProvider())()            Creates a default SyncProvider object. |

| **Method Summary** | |
| --- | --- |
| abstract  int | [**getDataSourceLock**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#getDataSourceLock())()            Returns the current data source lock severity level active in this SyncProvider implementation. |
| abstract  int | [**getProviderGrade**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#getProviderGrade())()            Returns a constant indicating the grade of synchronization a RowSet object can expect from this SyncProvider object. |
| abstract  [String](http://docs.google.com/java/lang/String.html) | [**getProviderID**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#getProviderID())()            Returns the unique identifier for this SyncProvider object. |
| abstract  [RowSetReader](http://docs.google.com/javax/sql/RowSetReader.html) | [**getRowSetReader**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#getRowSetReader())()            Returns a javax.sql.RowSetReader object, which can be used to populate a RowSet object with data. |
| abstract  [RowSetWriter](http://docs.google.com/javax/sql/RowSetWriter.html) | [**getRowSetWriter**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#getRowSetWriter())()            Returns a javax.sql.RowSetWriter object, which can be used to write a RowSet object's data back to the underlying data source. |
| abstract  [String](http://docs.google.com/java/lang/String.html) | [**getVendor**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#getVendor())()            Returns the vendor name of this SyncProvider instance |
| abstract  [String](http://docs.google.com/java/lang/String.html) | [**getVersion**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#getVersion())()            Returns the release version of this SyncProvider instance. |
| abstract  void | [**setDataSourceLock**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#setDataSourceLock(int))(int datasource\_lock)            Sets a lock on the underlying data source at the level indicated by *datasource\_lock*. |
| abstract  int | [**supportsUpdatableView**](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#supportsUpdatableView())()            Returns whether this SyncProvider implementation can perform synchronization between a RowSet object and the SQL VIEW in the data source from which the RowSet object got its data. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

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

### GRADE\_NONE

public static int **GRADE\_NONE**

Indicates that no synchronization with the originating data source is provided. A SyncProvider implementation returning this grade will simply attempt to write updates in the RowSet object to the underlying data source without checking the validity of any data.

### GRADE\_CHECK\_MODIFIED\_AT\_COMMIT

public static int **GRADE\_CHECK\_MODIFIED\_AT\_COMMIT**

Indicates a low level optimistic synchronization grade with respect to the originating data source. A SyncProvider implementation returning this grade will check only rows that have changed.

### GRADE\_CHECK\_ALL\_AT\_COMMIT

public static int **GRADE\_CHECK\_ALL\_AT\_COMMIT**

Indicates a high level optimistic synchronization grade with respect to the originating data source. A SyncProvider implementation returning this grade will check all rows, including rows that have not changed.

### GRADE\_LOCK\_WHEN\_MODIFIED

public static int **GRADE\_LOCK\_WHEN\_MODIFIED**

Indicates a pessimistic synchronization grade with respect to the originating data source. A SyncProvider implementation returning this grade will lock the row in the originating data source.

### GRADE\_LOCK\_WHEN\_LOADED

public static int **GRADE\_LOCK\_WHEN\_LOADED**

Indicates the most pessimistic synchronization grade with respect to the originating data source. A SyncProvider implementation returning this grade will lock the entire view and/or table affected by the original statement used to populate a RowSet object.

### DATASOURCE\_NO\_LOCK

public static int **DATASOURCE\_NO\_LOCK**

Indicates that no locks remain on the originating data source. This is the default lock setting for all SyncProvider implementations unless otherwise directed by a RowSet object.

### DATASOURCE\_ROW\_LOCK

public static int **DATASOURCE\_ROW\_LOCK**

Indicates that a lock is placed on the rows that are touched by the original SQL statement used to populate the RowSet object that is using this SyncProvider object.

### DATASOURCE\_TABLE\_LOCK

public static int **DATASOURCE\_TABLE\_LOCK**

Indicates that a lock is placed on all tables that are touched by the original SQL statement used to populate the RowSet object that is using this SyncProvider object.

### DATASOURCE\_DB\_LOCK

public static int **DATASOURCE\_DB\_LOCK**

Indicates that a lock is placed on the entire data source that is the source of data for the RowSet object that is using this SyncProvider object.

### UPDATABLE\_VIEW\_SYNC

public static int **UPDATABLE\_VIEW\_SYNC**

Indicates that a SyncProvider implementation supports synchronization between a RowSet object and the SQL VIEW used to populate it.

### NONUPDATABLE\_VIEW\_SYNC

public static int **NONUPDATABLE\_VIEW\_SYNC**

Indicates that a SyncProvider implementation does **not** support synchronization between a RowSet object and the SQL VIEW used to populate it.

| **Constructor Detail** |
| --- |

### SyncProvider

public **SyncProvider**()

Creates a default SyncProvider object.

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

### getProviderID

public abstract [String](http://docs.google.com/java/lang/String.html) **getProviderID**()

Returns the unique identifier for this SyncProvider object.

**Returns:**a String object with the fully qualified class name of this SyncProvider object

### getRowSetReader

public abstract [RowSetReader](http://docs.google.com/javax/sql/RowSetReader.html) **getRowSetReader**()

Returns a javax.sql.RowSetReader object, which can be used to populate a RowSet object with data.

**Returns:**a javax.sql.RowSetReader object

### getRowSetWriter

public abstract [RowSetWriter](http://docs.google.com/javax/sql/RowSetWriter.html) **getRowSetWriter**()

Returns a javax.sql.RowSetWriter object, which can be used to write a RowSet object's data back to the underlying data source.

**Returns:**a javax.sql.RowSetWriter object

### getProviderGrade

public abstract int **getProviderGrade**()

Returns a constant indicating the grade of synchronization a RowSet object can expect from this SyncProvider object.

**Returns:**an int that is one of the following constants: SyncProvider.GRADE\_NONE, SyncProvider.GRADE\_CHECK\_MODIFIED\_AT\_COMMIT, SyncProvider.GRADE\_CHECK\_ALL\_AT\_COMMIT, SyncProvider.GRADE\_LOCK\_WHEN\_MODIFIED, SyncProvider.GRADE\_LOCK\_WHEN\_LOADED

### setDataSourceLock

public abstract void **setDataSourceLock**(int datasource\_lock)  
 throws [SyncProviderException](http://docs.google.com/javax/sql/rowset/spi/SyncProviderException.html)

Sets a lock on the underlying data source at the level indicated by *datasource\_lock*. This should cause the SyncProvider to adjust its behavior by increasing or decreasing the level of optimism it provides for a successful synchronization.

**Parameters:**datasource\_lock - one of the following constants indicating the severity level of data source lock required:

SyncProvider.DATASOURCE\_NO\_LOCK,  
 SyncProvider.DATASOURCE\_ROW\_LOCK,  
 SyncProvider.DATASOURCE\_TABLE\_LOCK,  
 SyncProvider.DATASOURCE\_DB\_LOCK,

**Throws:** [SyncProviderException](http://docs.google.com/javax/sql/rowset/spi/SyncProviderException.html) - if an unsupported data source locking level is set.**See Also:**[getDataSourceLock()](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#getDataSourceLock())

### getDataSourceLock

public abstract int **getDataSourceLock**()  
 throws [SyncProviderException](http://docs.google.com/javax/sql/rowset/spi/SyncProviderException.html)

Returns the current data source lock severity level active in this SyncProvider implementation.

**Returns:**a constant indicating the current level of data source lock active in this SyncProvider object; one of the following:

SyncProvider.DATASOURCE\_NO\_LOCK,  
 SyncProvider.DATASOURCE\_ROW\_LOCK,  
 SyncProvider.DATASOURCE\_TABLE\_LOCK,  
 SyncProvider.DATASOURCE\_DB\_LOCK

**Throws:** SyncProviderExceptiom - if an error occurs determining the data source locking level. [SyncProviderException](http://docs.google.com/javax/sql/rowset/spi/SyncProviderException.html)**See Also:**[setDataSourceLock(int)](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html#setDataSourceLock(int))

### supportsUpdatableView

public abstract int **supportsUpdatableView**()

Returns whether this SyncProvider implementation can perform synchronization between a RowSet object and the SQL VIEW in the data source from which the RowSet object got its data.

**Returns:**an int saying whether this SyncProvider object supports updating an SQL VIEW; one of the following: SyncProvider.UPDATABLE\_VIEW\_SYNC, SyncProvider.NONUPDATABLE\_VIEW\_SYNC

### getVersion

public abstract [String](http://docs.google.com/java/lang/String.html) **getVersion**()

Returns the release version of this SyncProvider instance.

**Returns:**a String detailing the release version of the SyncProvider implementation

### getVendor

public abstract [String](http://docs.google.com/java/lang/String.html) **getVendor**()

Returns the vendor name of this SyncProvider instance

**Returns:**a String detailing the vendor name of this SyncProvider implementation

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SyncProvider.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/SyncFactoryException.html)   [**NEXT CLASS**](http://docs.google.com/javax/sql/rowset/spi/SyncProviderException.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sql/rowset/spi/SyncProvider.html)    [**NO FRAMES**](http://docs.google.com/SyncProvider.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#3j2qqm3) | [METHOD](#4i7ojhp) |

[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).