Progress DataDirect *for* JDBC for Salesforce 6.0 Compatibility FAQ

This FAQ highlights changes in behavior between the Progress DataDirect *for* JDBC for Salesforce 5.1 and 6.0 drivers. These changes include differences in how Salesforce objects are exposed to applications, and differences in how some SQL queries may be handled. Review these questions and take appropriate action to ensure the compatibility of the driver with your environment, as you upgrade from the 5.1 version of the driver to the 6.0 version.

Have any requirements changed for the driver?

No. However, the 5.1 driver was updated to require a Java SE 7 or higher JVM to comply with enhancements to Salesforce security standards. This requirement also applies to the 6.0 driver.

Has the data source class changed for the 6.0 driver?

The 6.0 driver provides only one data source class: `com.ddtek.jdbcx.sforce.SForceDataSource`. This data source class supports all JDBC specifications. In previous releases, `com.ddtek.jdbcx.sforce.SForceDataSource` supported 3.*x* and earlier specifications, and `com.ddtek.jdbcx.sforce.SForceDataSource40` supported 4.0 and later specifications. For the 6.0 release, `com.ddtek.jdbcx.sforce.SForceDataSource40` has been deprecated.

Does your connection string or JDBC data source specify a configuration file with the DatabaseName (or Database) connection property?

If so, you should update your connection string or data source to specify the configuration file with the SchemaMap property. This configuration file contains a relational map of your data used to support SQL queries against Salesforce. The configuration file is in XML format and can be shared across client machines.

If your connection string or JDBC data source does not specify a configuration file with the DatabaseName (or Database) connection property, then no further action is required. When your application connects to Salesforce, the driver will create this file if it does not exist in a default location. However, you can specify the fully qualified path of the configuration file with the SchemaMap property if you prefer. This configuration file contains a map of your data used support SQL queries against Salesforce. The configuration file is in XML format and can be shared across client machines.

Are there any changes in how the driver manages cached data?

The RefreshDirtyCache property has been deprecated for the 6.0 release. Now, for every fetch operation, the 6.0 driver refreshes the cached object to pick up changes made to tables and rows.

Can I expect the same query results with the 6.0 driver as those I received with the 5.1 driver?

For the 6.0 release, the driver's SQL engine has been upgraded. Consequently, there are some differences in how the 6.0 driver handles SQL queries compared to the 5.1 driver. Refer to this SQL engine upgrade document for details.

You might also notice that the 6.0 driver uses more API calls for some correlated join queries compared to the 5.1 driver. The 6.0 driver returns the entire first table and then sends multiple requests to fetch corresponding rows for the second table, whereas the 5.1 driver returns both tables and then processes the query locally.

Does the driver support the Salesforce Bulk API and PK chunking?

With the 6.0 release, the driver supports bulk fetch operations, in addition to bulk inserts, updates, and deletes, via the Salesforce Bulk API. Support for PK chunking has also been added for this release. Bulk fetch can be enabled and configured with the BulkFetchThreshold, EnableBulkFetch, EnablePKChunking, and PKChunkSize connection properties. See the user's guide for details.

Has the driver been modified in other ways that might affect the execution of bulk operations?

Yes. Based on customer feedback, the driver is enabled by default to use the Salesforce Bulk API for inserts, updates, and deletes. To disable this behavior, set EnableBulkLoad to `false` in your connection string or JDBC data source. Alternatively, you can configure this behavior with the BulkLoadThreshold property. See the user's guide for details.

Can the driver handle multiple web sessions?

Yes. The 6.0 driver supports simultaneous multiple sessions. You can set the WSPoolSize property to specify the maximum number of sessions allowed. See the user's guide for details.

Have there been other changes to the driver that affect how web sessions are handled?

Yes. The default value for the StmtCallLimit connection property has been updated to 100. By default, the driver can make a maximum of 100 web service calls when executing any single SQL statement or metadata query. To revert to the 5.1 default behavior, set StmtCallLimit to `20` in your connection string or JDBC data source.

NOTE: The `STMT_CALL_LIMIT` attribute in an `ALTER SESSION` statement overrides this setting. For example, `ALTER SESSION SET STMT_CALL_LIMIT=10`.

Have any changes been made to how the driver exposes custom objects?

Yes. Based on customer feedback, the default behavior of the driver has been modified to include the __c suffix with custom table and column names when mapping the Salesforce data model. To revert to the 5.1 default behavior, set CustomSuffix to `strip` with the ConfigOptions connection property. Setting CustomSuffix to `strip` removes the _c suffix from custom table and column names.

Have any changes been made to how the driver exposes system fields?

Yes. Based on customer feedback, the default behavior of the driver has been modified to expose the names of system fields as they exist in the Salesforce data model. In other words, the 6.0 driver does not change the names of system fields when mapping the Salesforce data model. To revert to the 5.1 default behavior, set MapSystemColumnNames to `1` with the ConfigOptions connection property. When MapSystemColumnNames is set to `1`, the driver modifies system column names such that the `Id` field is mapped as `ROWID` and the remaining system columns are prefixed with `SYS_` (for example, the `Name` field would be mapped as `SYS_NAME`).

NOTE: All identifiers are mapped as uppercase by default in the 5.1 and 6.0 drivers. See "UppercaseIdentifiers (Configuration Option)" in the user's guide for information on changing this behavior.

Does the driver expose Salesforce audit fields and the master record ID field?

Yes. Based on customer feedback, the default behavior of the driver has been modified to expose audit fields and the master record ID field. To revert to the 5.1 default behavior, set AuditColumns to none with the ConfigOptions connection property. Setting AuditColumns to none hides audit fields and the master record ID field to client applications.