SQL-SALES

User Guide

&

Technical Reference

Version 25.1.1.2

CONTENTS

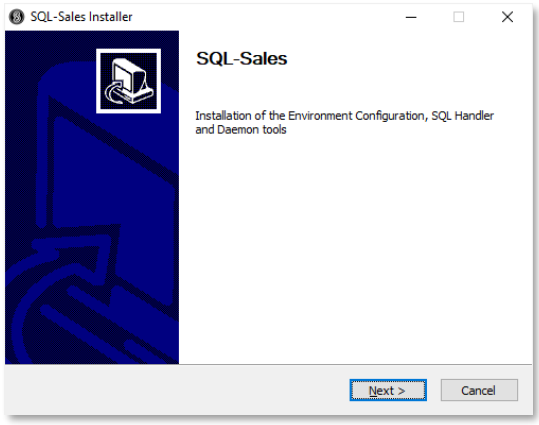
INSTALLATION

Installing SQL-Sales to your SQL Server from the provided "SQLSalesInstaller.exe" is straight-forward with minimal inputs required to get going on working with Salesforce data directly in your SQL Server.

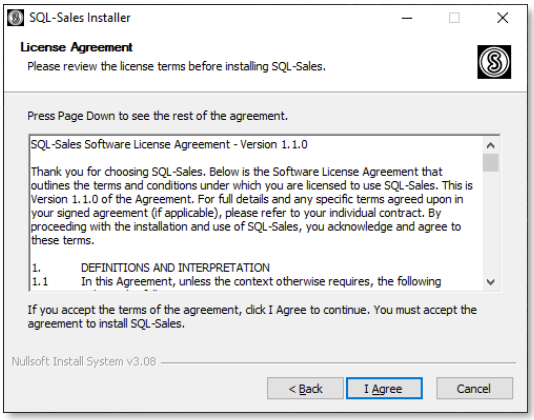
Note, you will also need to install the SQL-Sales Managed Package, see here

Once the package is installed, you can immediately start using SQL-Sales by connecting in the Configuration tool (you are about to install) with Username-Password (SOAP or REST api). However if you wish to connect with OAuth 2.0 (REST api) you will need to make the configuration changes, here

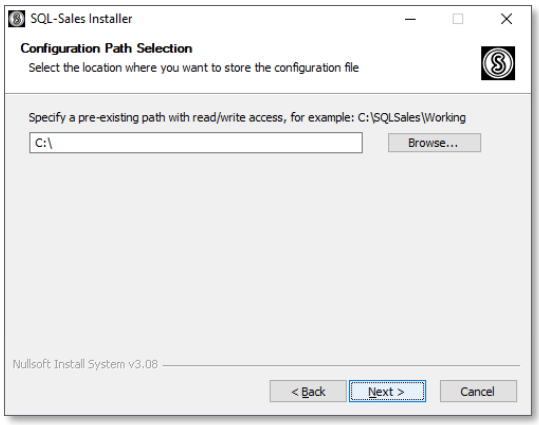
SQLSalesInstaller.exe

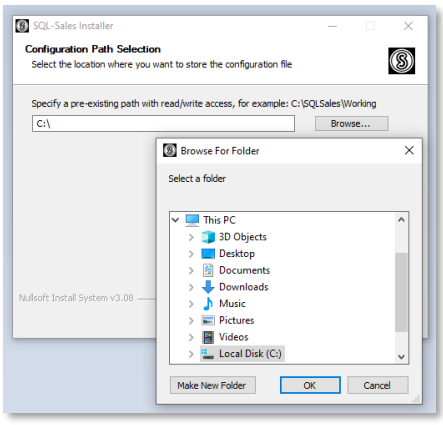


Start here



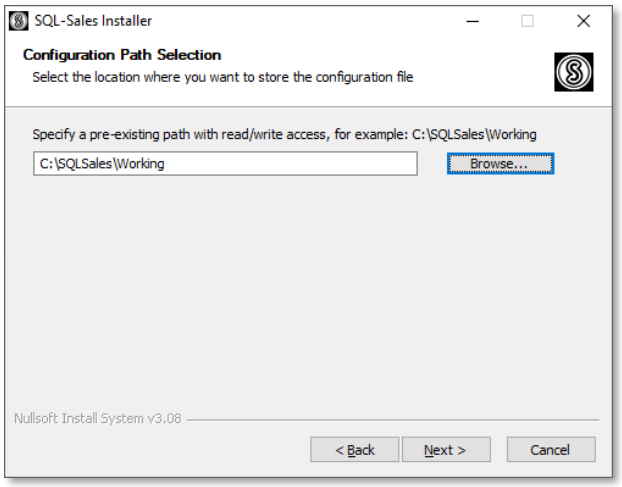
Licence Agreement





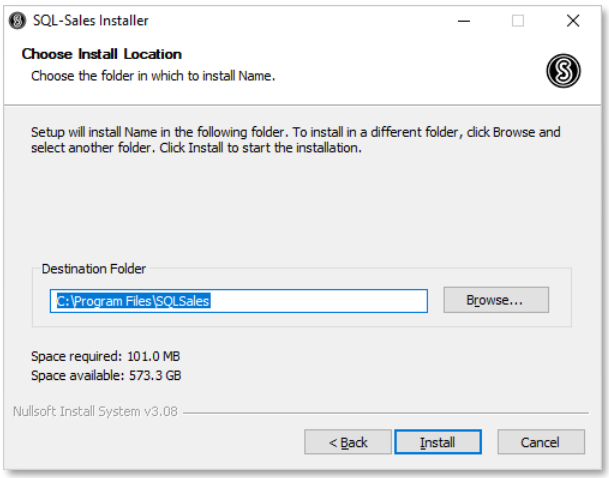
SQL-Sales

By default the location of the SQL-Sales config will be to root drive where your Windows OS is installed (typically C:\)

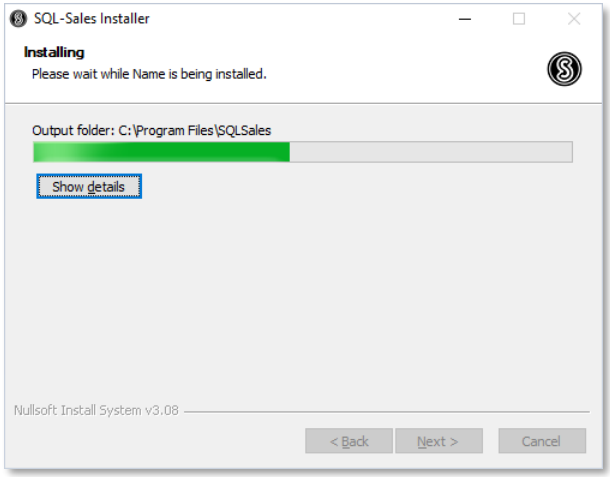


Example of a customised configuration path

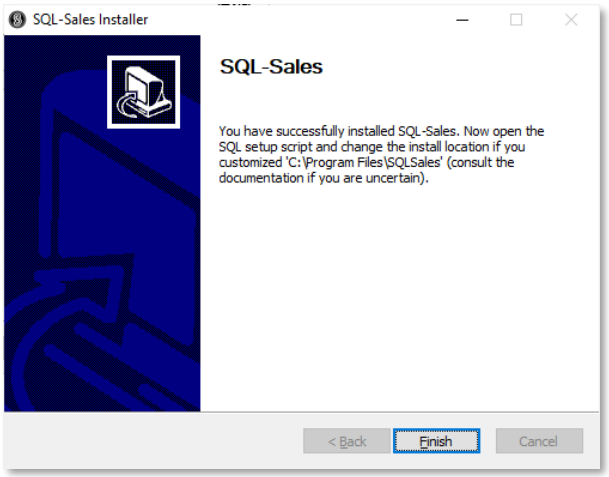
Customise this as required (usually to a more appropriate folder on the SQL Server such as C:\SQLSales\Working)



Generally, accept the default installation location, or customise as required



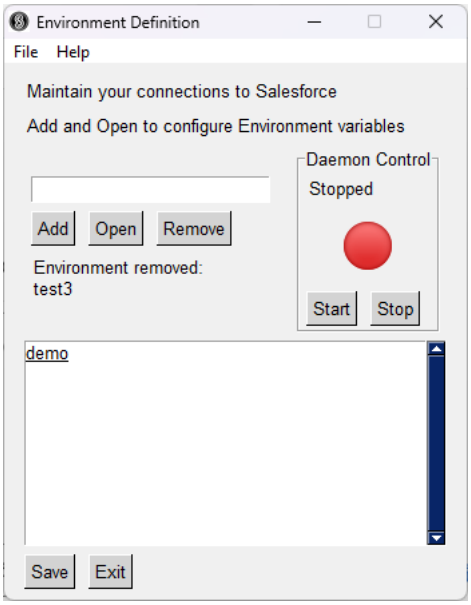
Installation running…



You're all set, now Open from your SQL Server Start Menu

Select "SQL-Sales Config" to open the SQL Sales Environment Definition tool.

In the guide, the Environment "DEMO" has been created. Select your Environment name and click "Open".



Daemon Status

The SQL-Sales Daemon is started automatically the first time it is called (by running one of the stored procedures[1]). It can be controlled from a user's client SSMS by running:

```
exec ss_Admin 'STOP'
```

against any SQL-Sales enabled database.

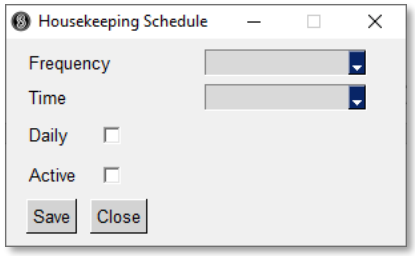
In normal use, the stored procedure will ensure the Daemon is kept running , however the Start/Stop buttons allow for quick setup smoke testing on the installation SQL Server, removing the need to run a stored procedure.

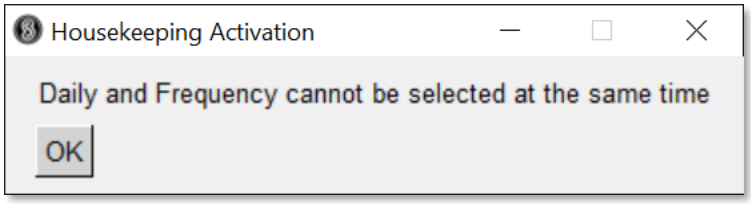
[1] See "Enabling Database" section

Housekeeping Schedule

File>>Housekeeping

This feature instructs SQL Sales to perform auto housekeeping tasks at a set time either Daily or Weekly. It is recommended (but not mandatory) that you setup a schedule, for example 3am daily. Pause an existing schedule, if required, by unchecking "Active".





Attribute	Notes
Frequency	Monday-Sunday picklist (if a daily schedule is required and not a set day per week, either don't select a day or pick the empty day at the top of the list).
Time	Choose an HH:MM minute in the range 00:00 to 23:59
Daily	Checkbox (if daily is required, do not select a day in Frequency, else you'll encounter the validation message shown above)
Active	Checkbox

Environment Setup

Opening a given Environment from the definition window produces the second main window below where you configure a specific Environment, refer to the attributes reference that follows for detailed guidance, although at a minimum, to get started immediately, you'll need to enter:

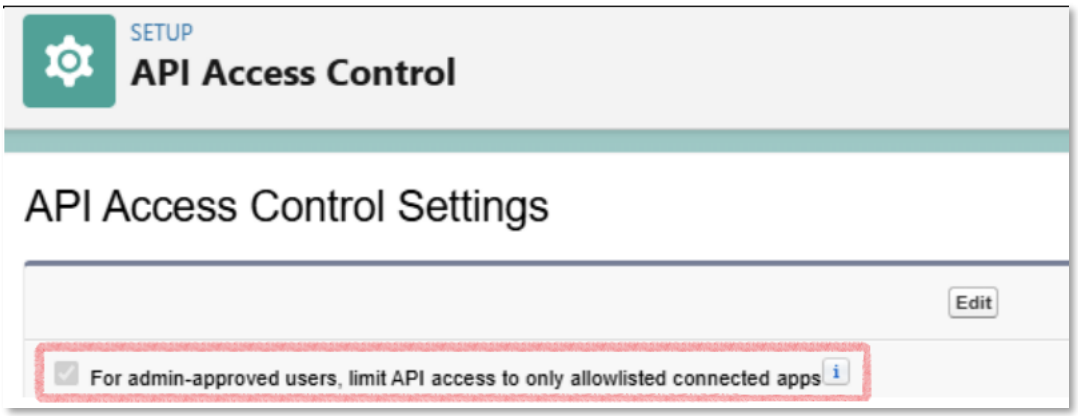
- Sandbox (checkbox, hence tick/not ticked)
- Username, Password, Security Token
- Working Path

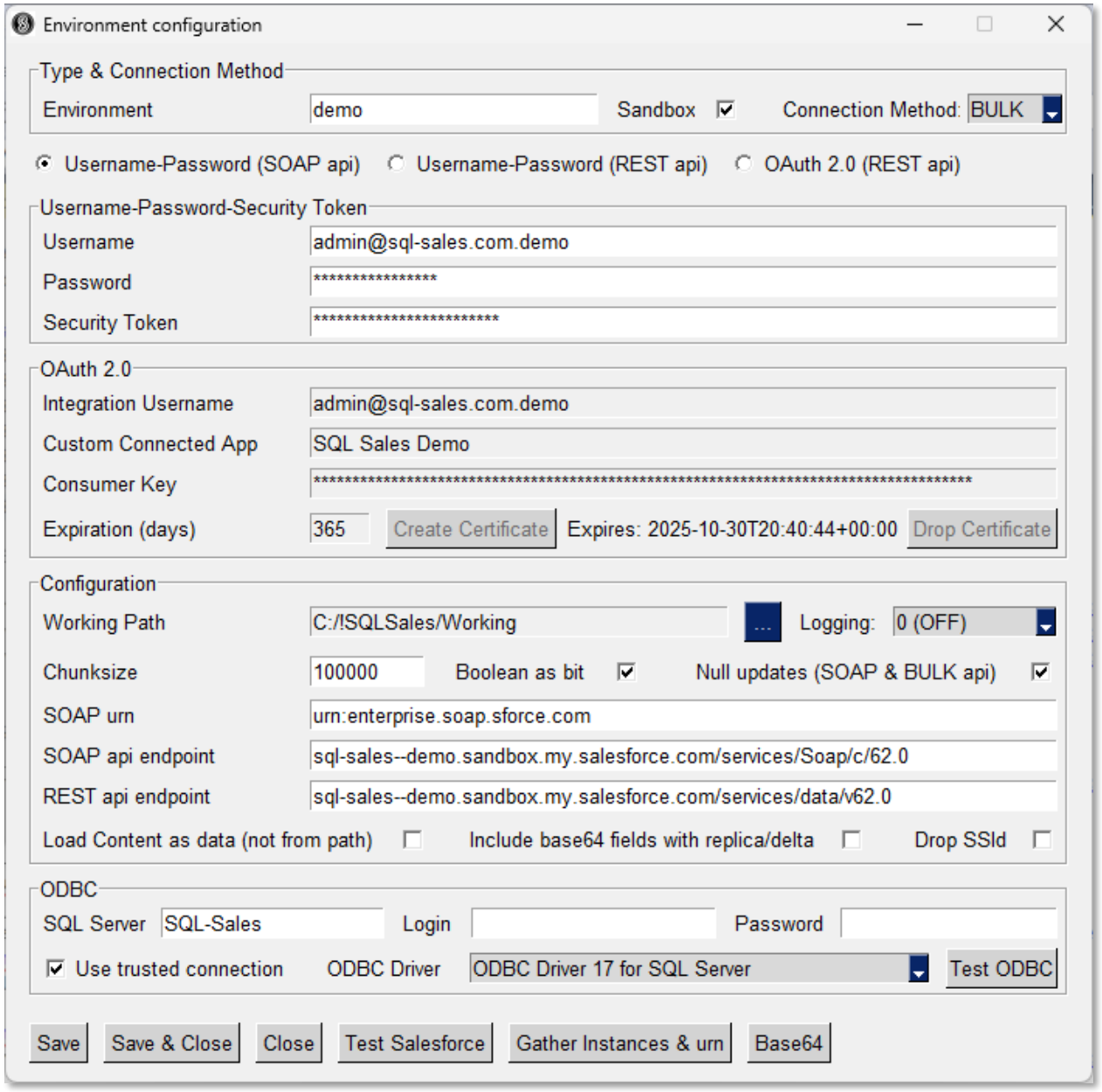
Note, if the Org you are installing to has this security setting enabled (which has to be requested to SF to be enabled, so is not an available feature by default):

Security >> API Access Control >> API Access Control Settings >> "**For admin-approved users, limit API access to only allowlisted connected apps**" = TRUE - this effectively means you must connect via the OAuth2.0 option. If you try with Username-Password-Security Token (SOAP or REST) you will encounter this message when testing the connection:

CLIENT_NOT_ACCESSIBLE_FOR_USER: Sorry, your administrator has blocked access to this client.

The SQL-Sales managed connected app is allowlisted but inherently OAuth 2.0. Therefore unless that setting can be unticked in your Org - if so, Username-Password-Security Token (SOAP or REST) can be used too.





Attribute	Notes
Environment	An Environment defines each unique connection to a Salesforce instance, whether that is a Sandbox or a Production Org, all connection work is done against this Name
Sandbox	Click this checkbox if your given Environment is a sandbox. Note, there are no licence restrictions imposed on accessing a sandbox, regardless of the Salesforce Org they belong to (i.e. even Orgs different to the one(s) you have licenced have no cost implication).
LogDate	Last log datetime

Attribute	Notes
Username	The Salesforce User used in the connection. This will be a Production username for your Production instance, or if you're defining a sandbox Environment, then a sandbox username.
Password	The Password of your username (text security obscured)
Security Token	The Security Token of your username (text security obscured)
Working Path	SQL Sales for the BULK Connection Method makes temporary use of a working directory on a drive available to the SQL Server. No data is held at rest and you are advised to check there is suitable spare capacity on the drive which hosts this directory.
Chunksize	Defaulted to 100000. SQL Sales will chunk internal data processing to this setting, separate to any api batchsize you may have specified for a given replication or data operation. Note, for certain heavier data operations, the specified Chunksize may be automatically lowered to achieve optimal data transfer between Salesforce and SQL Server.
Boolean as bit	Salesforce will by default return boolean fields as TRUE or FALSE, which SQL Server handles in a varchar(5) datatype. Working in SQL Server it is often preferable to work with a bit datatype.
Null updates (SOAP & BULK api)	By default when using the loading to Salesforce with the SOAP and BULK apis, you are not permitted to update a field with an null, this setting overrides that. REST by default does permit a null over-write hence not part of this setting.
DropSSId	When loading to Salesforce, via a SQL Server table, SQL Sales will define a primary key
Instance	This is your Salesforce instance url, click on "Gather Instance & urn" to populate for your given username. By default this will return your current Salesforce api version for your end point. This will always be the latest api version you are set to, however this can be customised to a specific older api version if that is relevant to your particular requirements. Customise by altering the api version number in the Instance string.
urn	This is your urn, which by default will return the urn for your Salesforce end point as defined by your username. Customise this if necessary for your particular requirements, typically you would expect to have one of these two: urn:enterprise.soap.sforce.com urn:partner.soap.sforce.com Generally, just accept the defaut returned for you
Load ContentVersion as data, not from file	When loading files to Salesforce, typically this is achieved by providing the path to a file located on a drive accessible by SQL Sales, in a UNC location. Alternatively, by checking this option, you can provide the file as a field in a SQL Server table (see detail in the ss_Loader stored procedure).

Attribute	Notes
Include base64 fields with replica/delta	By default, when replicating (via ss_Replica & ss_Delta), Salesforce fields which contain Content (base64) data are not included as they are binary data-heavy. If you require this data, check this box.
Connection Method (BULK/ODBC)	BULK (for the bulk text based method) ODBC (for ODBC). Typically BULK is quicker but you decide what works best for your situation.
Logging	Default is 0-OFF. The log file is created in the same location as the setup "Configuration Path Selection". Generally, logging is not necessary and certainly 1-DEBUG should only be used when diagnosing issues with SQL-Sales support 0-OFF 1-DEBUG 2-INFO 3-WARNING 4-ERROR 5-CRITICAL
ODBC: SQL Server	Applicable only for the ODBC method: specify your SQL Server, this will be the server on which you have installed SQL Sales.
ODBC: Use trusted connection	Applicable only for the ODBC method: If applicable to your setup and how you have installed SQL Sales (i.e. how you are running SQL Sales). Check this if Windows trusted connection is applicable to your use case.
ODBC: SQL Login	Applicable if connecting via a SQL Server login (i.e. not trusted connection).
ODBC: SQL Password	Applicable if connecting via a SQL Server login (i.e. not trusted connection). (text security obscured).
ODBC: Driver	List of available drivers available on your specified SQL Server.
Integration Username	The API or integration Username specified in the sub (subject) claim of the JWT token (when using OAuth 2.0 to authenticate and connect to Salesforce).
Custom Connected App	Each configured Environment in each installation of SQL-Sales will require a custom connected app setting up, enter the name here, for the given Environment (when using OAuth 2.0 to authenticate and connect to Salesforce). Note, SQL-Sales will only accept alphanumeric, space and underscore characters for the ConnectedApplication.Name.
Consumer Key	Consumer key, copied from the above custom connected app to authenticate and connect to Salesforce).
Expiration (days)	Enter a numeric value between 1 and 365. This inputs to the certificate validity period which you'll create, for passing to the custom connected app when setting it up to authenticate and connect to Salesforce).
"Create Certificate" button	This will generate and pass a new digital certificate to your clipboard, for you to save as a .pem file for uploading to your custom connected app. The certificate is not held at rest within SQL-Sales. This is required to authenticate and connect to Salesforce during the setup.

Attribute	Notes
"Drop Certificate" button	Instantly revoke an existing certificate.

Test Salesforce

validate your credentials and licence

Gather Instance & urn

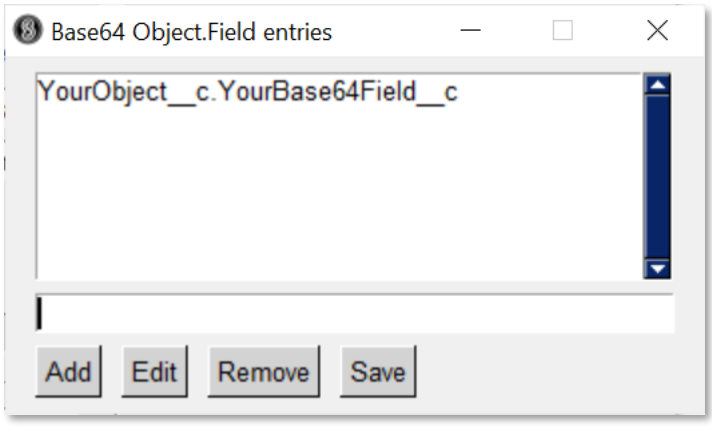
Click on this button to see your Instance and urn. Edit as needed (usually you should have no need to do this, but it can be helpful for more involved use cases. The example in the earlier "Environment" window is from our Demo developer Org.

Save | Save & Close

Save will save your settings, whereas Save & close will Save and close the Environment Configuration window, returning you to the first window, showing your list of Environments you have setup.

Base64

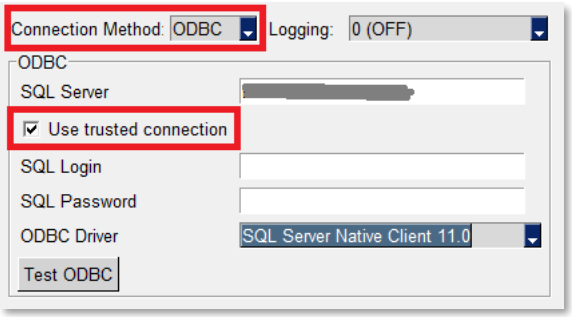
By default the following Objects and their Base64 field are included in what SQL Sales will action with the "Include base64 fields with replica/delta" option. Should you have a custom or additional requirement beyond this default state, clicking on "Base64" will open the window below where you can enter the field you wish to include in your replication.

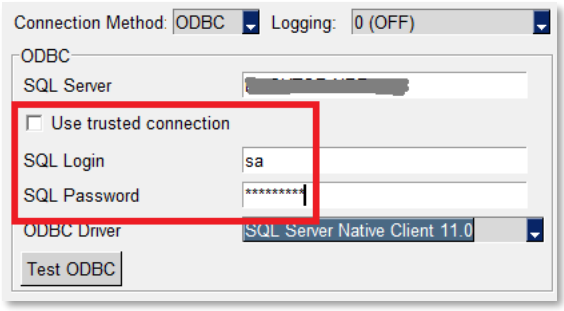


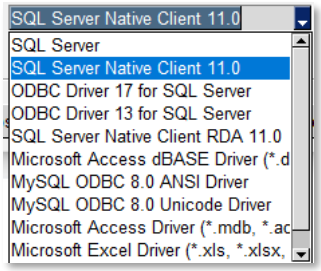
Included by Default
Attachment (Body)
ContentVersion (VersionData)
EmailCapture (RawMessage)
MailmergeTemplate (Body)
Scontrol (Binary)
ContentNote (Content)
Document (Body)
EventLogFile (LogFile)
MobileApplicationDetail (ApplicationBinary)
StaticResource (Body)

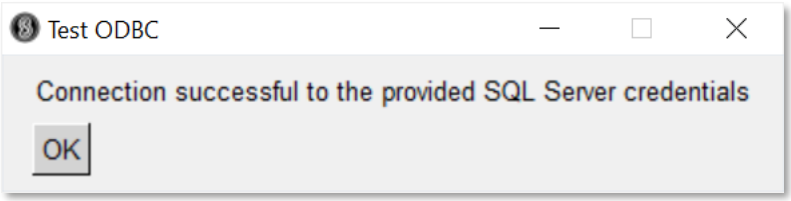
ODBC

Either connect using the trusted connection method or via direct SQL credentials (SQL Login and password). Select the driver available to you based on what is actually installed on your SQL Server (list shown below is purely illustrative).









Auto Close

The Configuration tool will auto close 12 hours after first opening or the last open, add, test or save event, whichever is greater.

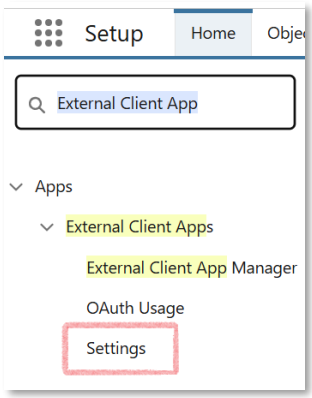
CONNECTING WITH OAUTH 2.0

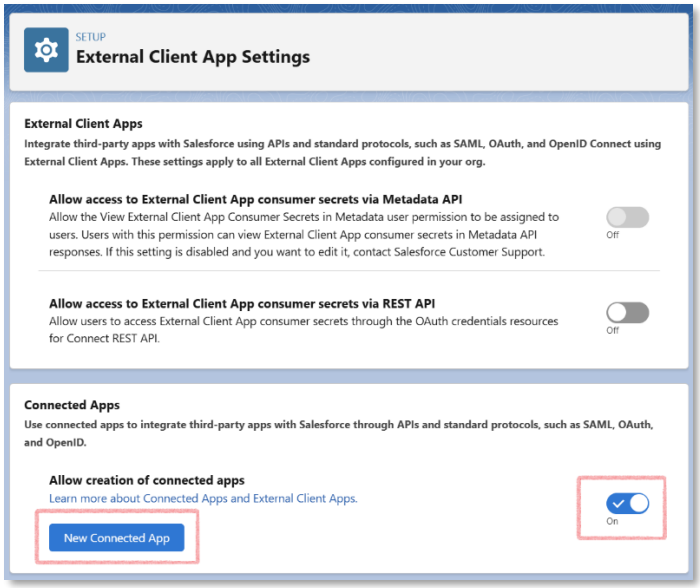
Note, you only need to create a custom Connected App for the OAuth 2.0 connection method. OAuth 2.0 provides increased levels of security and so may be required for some Customers or use cases, alternatively simply use the traditional Username-Password-Security Token (SOAP or REST api) connection method (i.e. which doesn't require a custom Connected App setting up).

All connection methods require the SQL-Sales Managed Package to be installed against the configured Environment (sandbox or Production instance).

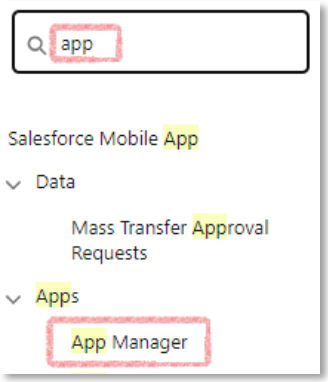
Create a Custom Connected App

In Setup type "External Client App", click on "Settings" as shown:





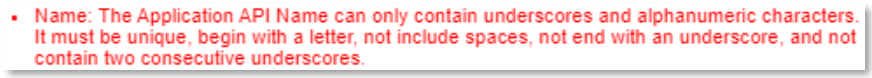
Enable "Connected Apps" and click on "New Connected App". You are about to create a new Connected App. To subsequently find your created connected app, type "app" in the setup menu, click on "App Manager"

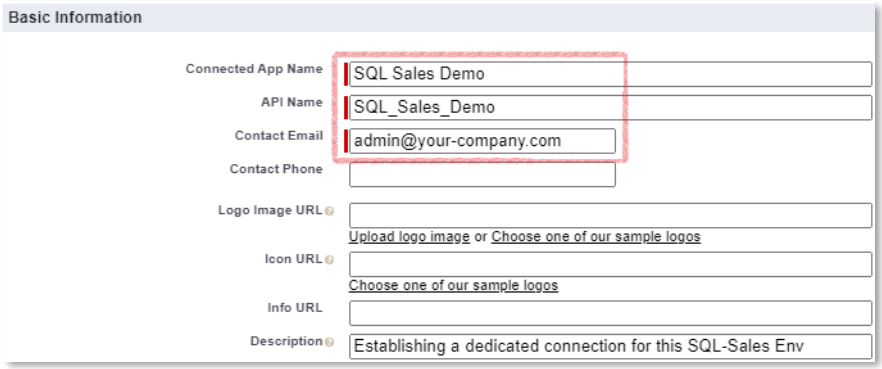


Step 1 – New Connected App

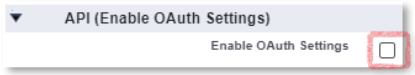
Input field	Notes
Connected App Name	Enter a suitable name for this Salesforce instance/sandbox, this is what you will eventually enter into the SQL-Sales Environment Configuration "Custom Connected App" input box. Note, SQL-Sales will only accept alphanumeric, space and underscore characters for the ConnectedApplication.Name
API Name	Salesforce will auto-populate based on the above name
Contact Email	Enter a suitable email for your use case

Note, there are data entry rules on the Custom App Name, specifically: *"The Application API Name can only contain underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores."* Which governs what can be created in the Name field (although spaces are permitted in the name field).



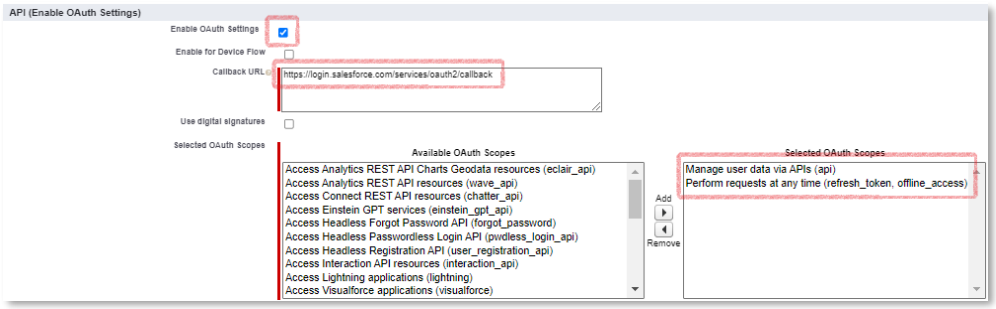


Step 2 - Click "Enable OAuth Settings

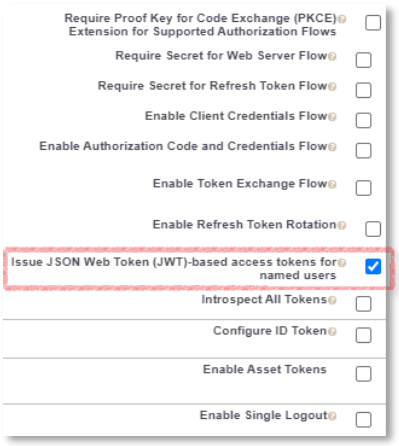


Input field	Notes
Enable OAuth Settings	Tick the checkbox

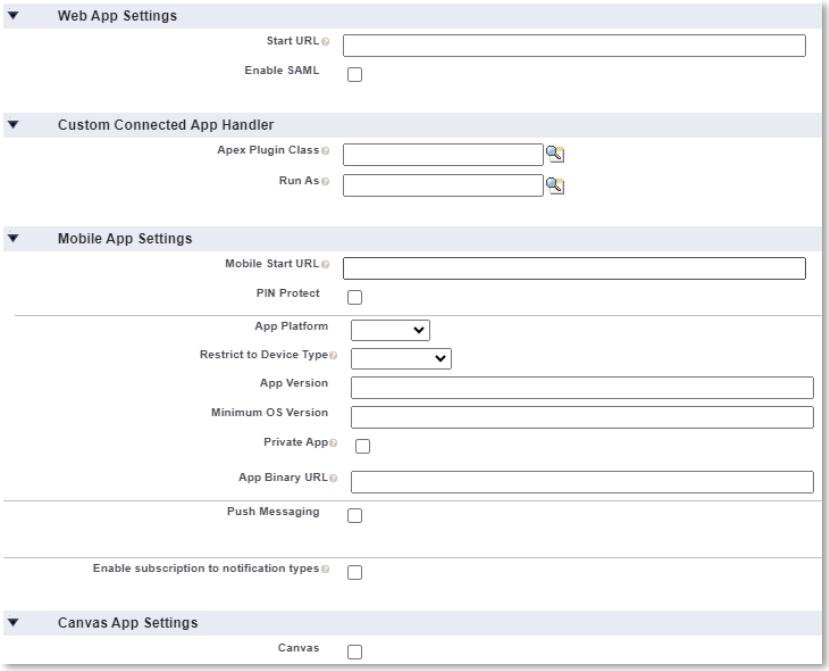
Input field	Notes
Use digital signatures	This is not actually referenced in the Connected App settings used by SQL-Sales, however it is a mandatory fill - entering the suggested default is fine as it does nothing functionally: https://login.salesforce.com/services/oauth2/callback
Use digital signatures	Revisited later in this article, once we have generated a self signed certificate within SQL-Sales. For now ignore.
Selected OAuth Scopes	Select only: Manage user data via APIs (api) Perform requests at any time (refresh_token, offline_access)



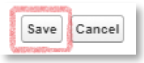
Step 3 - The only OAuth 2.0 option to tick is: "**Issue JSON Web Token (JWT)-based access tokens for named users**". SQL-Sales establishes and maintains connections to Salesforce only through JWT (tokens). All other options, even ones which may be ticked by default are to be left unticked / False.



Step 4 - For the avoidance of doubt, no other options are to be enabled, no WebApp Settings; no Custom Connected App Handler; no Mobile App Settings and no Canvas App Settings.



Step 5 - Save (we will return to complete the certificate upload later)

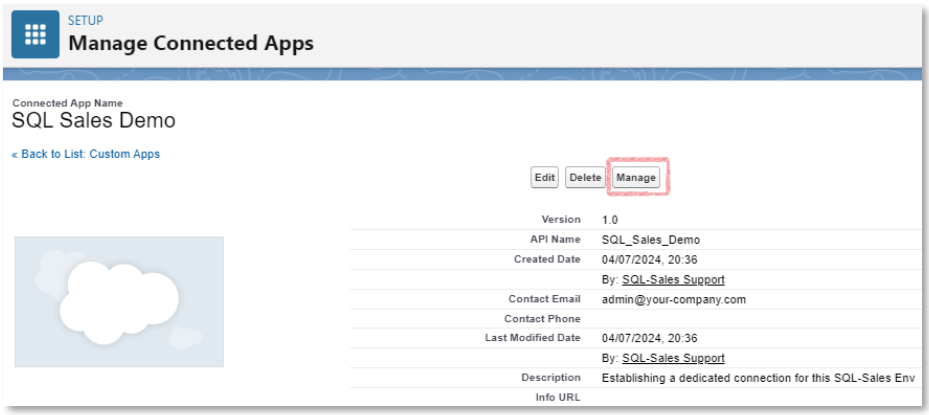


Step 6 - Click "Continue" at the notification prompt below, following the Save:

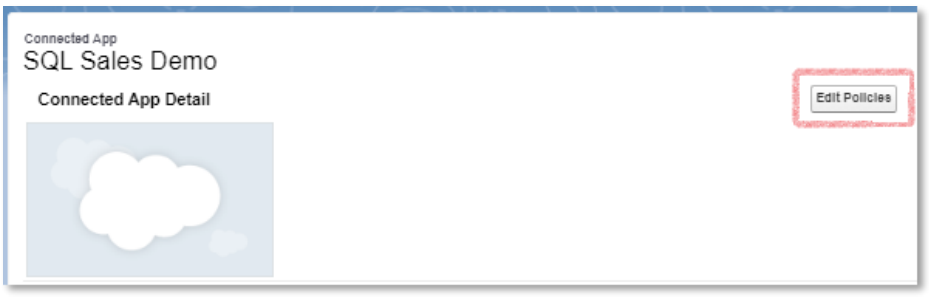
Manage Connected Apps

Step 7 - Following the Save>>Continue you'll be taken to the "Manage Connected App" window.

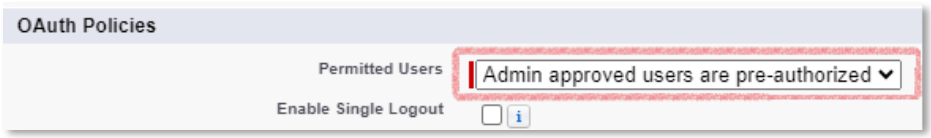
Click "Manage"



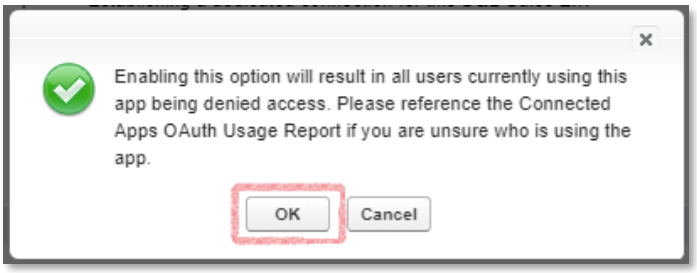
Step 8 - Click "Edit Policies"



Step 9 - In "Permitted Users" select "Admin approved users are pre-authorized"

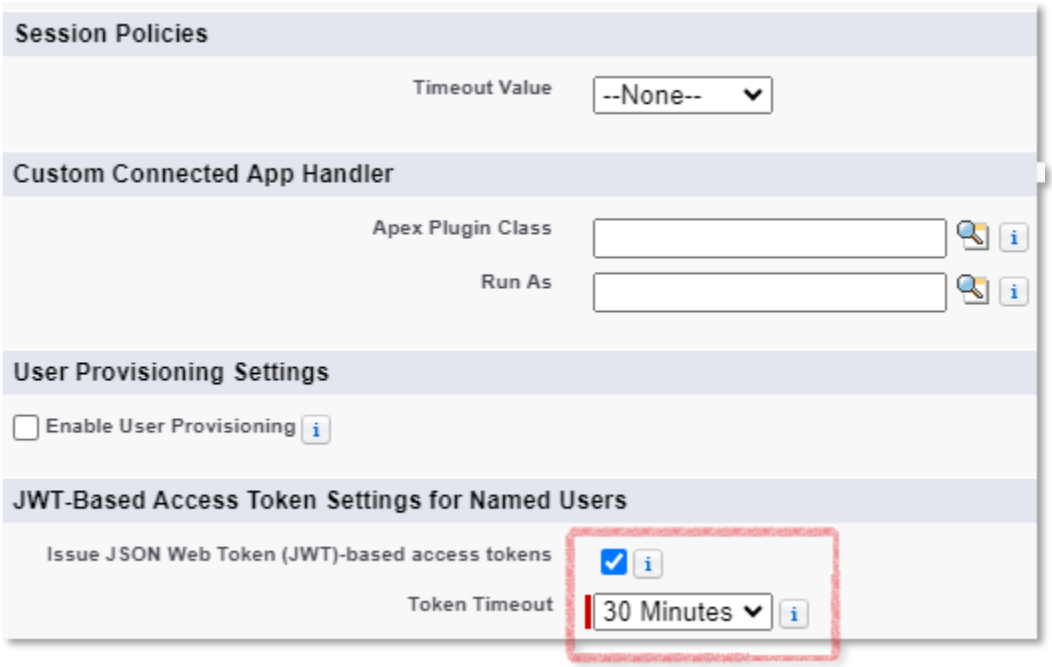


You will receive the confirmation below, click "OK".

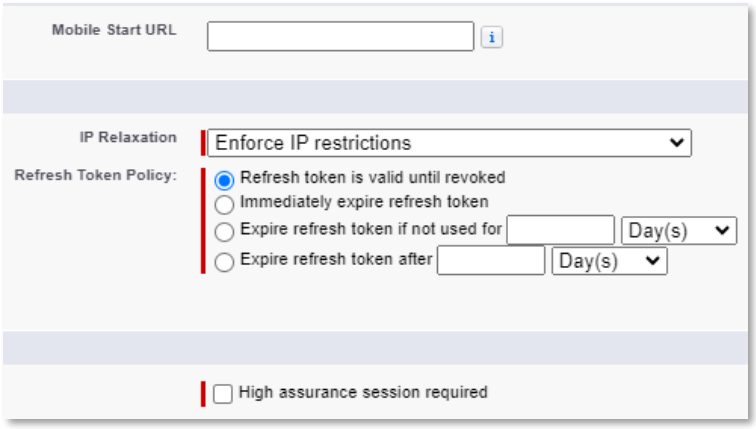


Step 10 - Ensure all options below are not entered / enabled - with the exception of:

Input field	Notes
Issue JSON Web Token (JWT)-based access tokens	Tick the checkbox
Token Timeout	Select 30 Minutes (Note, SQL-Sales at run time will validate that 30 Minutes is the configured setting).



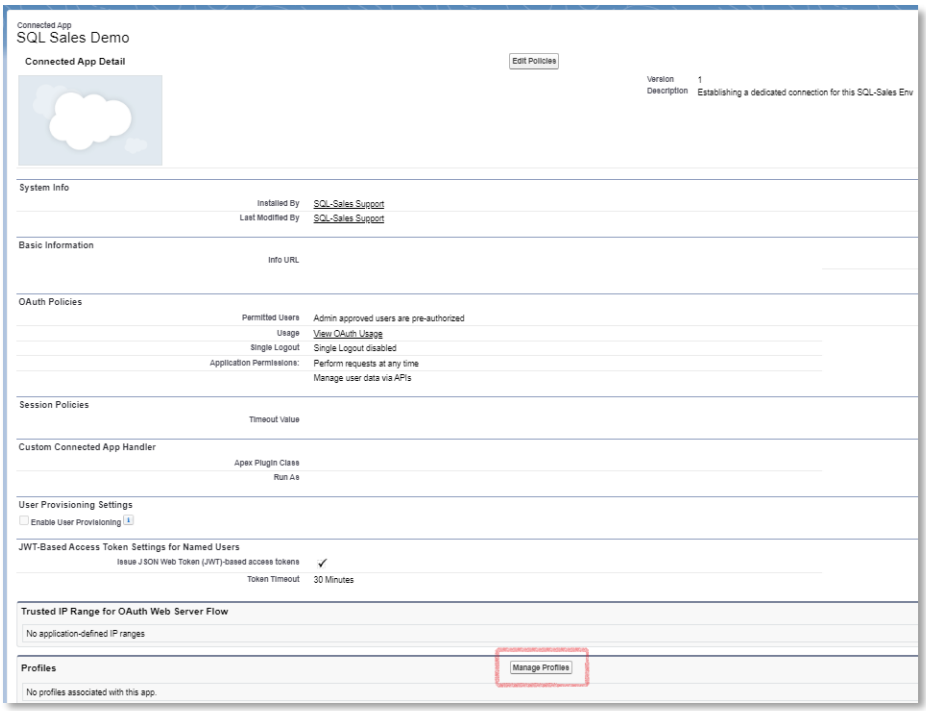
Step 11 - The remainder policy options should be defaulted as shown, if not ensure you have the settings below:



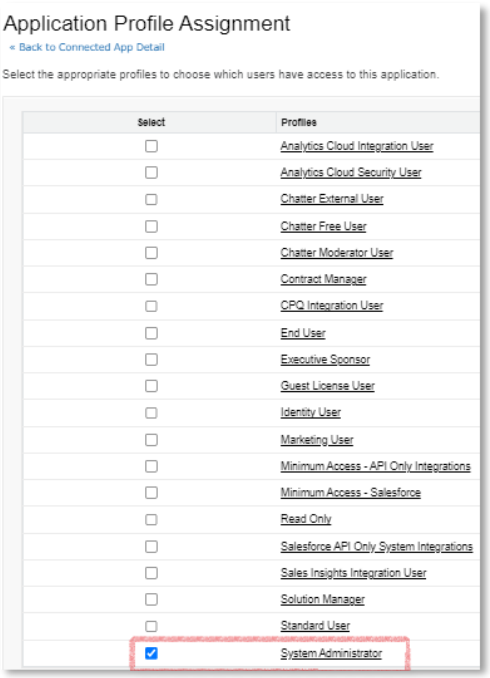
Step 12 - Save the "edit policies" section

Step 13 - Connected App Detail - Manage Profiles

You'll be returned to the Connected App Detail window. Click on the "Manage Profiles" button:

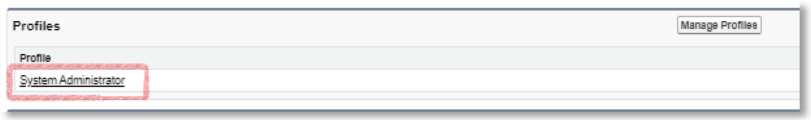


Step 14 - Select the Profile of the Username you will be defining in SQL-Sales as the nominated OAuth Username (the Integration Username).

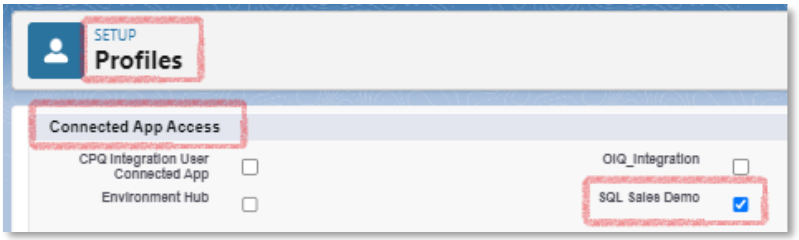


Click Save

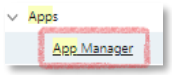
Your Profile will be listed as below:



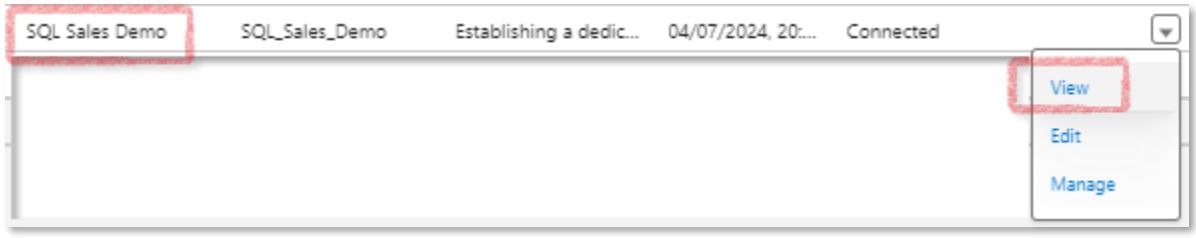
Alternatively, from directly within the given Profile tick your created Connected App from there, it has the same effect as the above. The below, in our example is the "System Administrator" Profile.



Step 15 - Now return to the "App Manager"

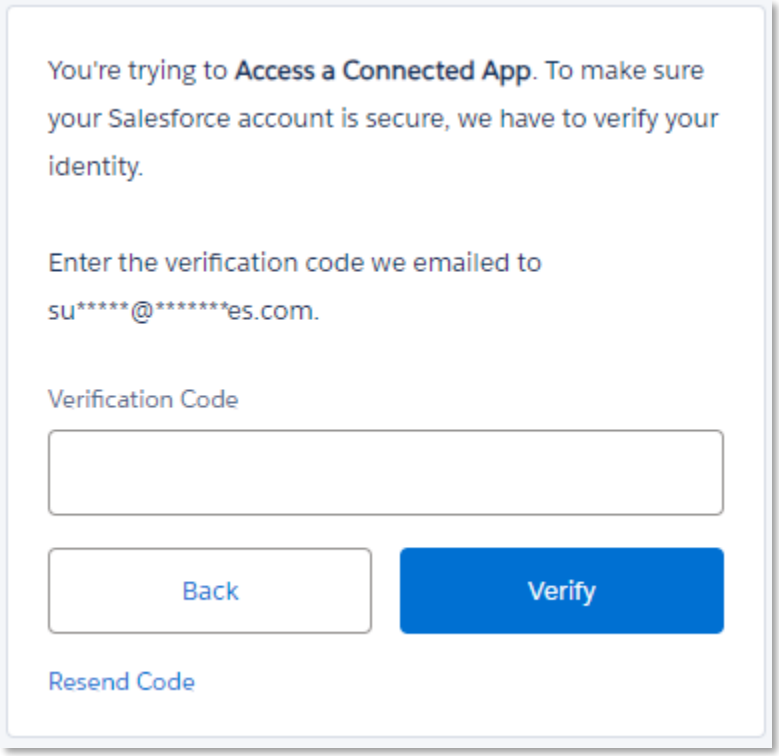


For your Connected App, choose "View"

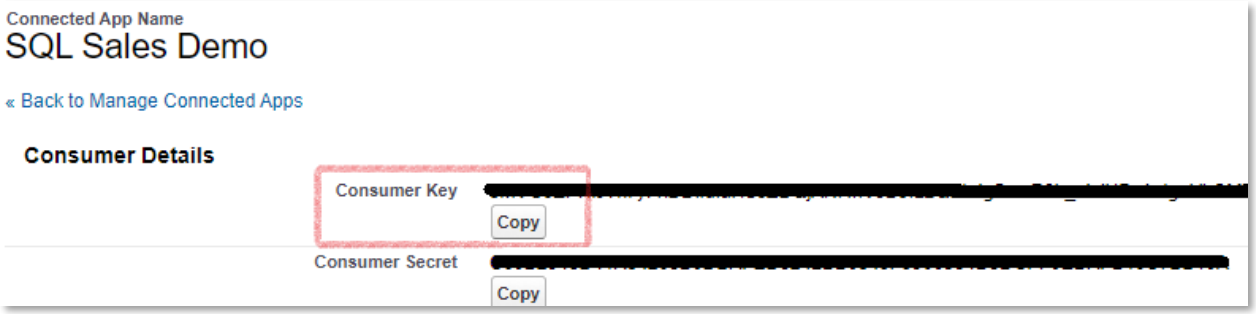


Step 16 - Click "Manage consumer Details"

Note, this will trigger a validation/security verification code request to your email

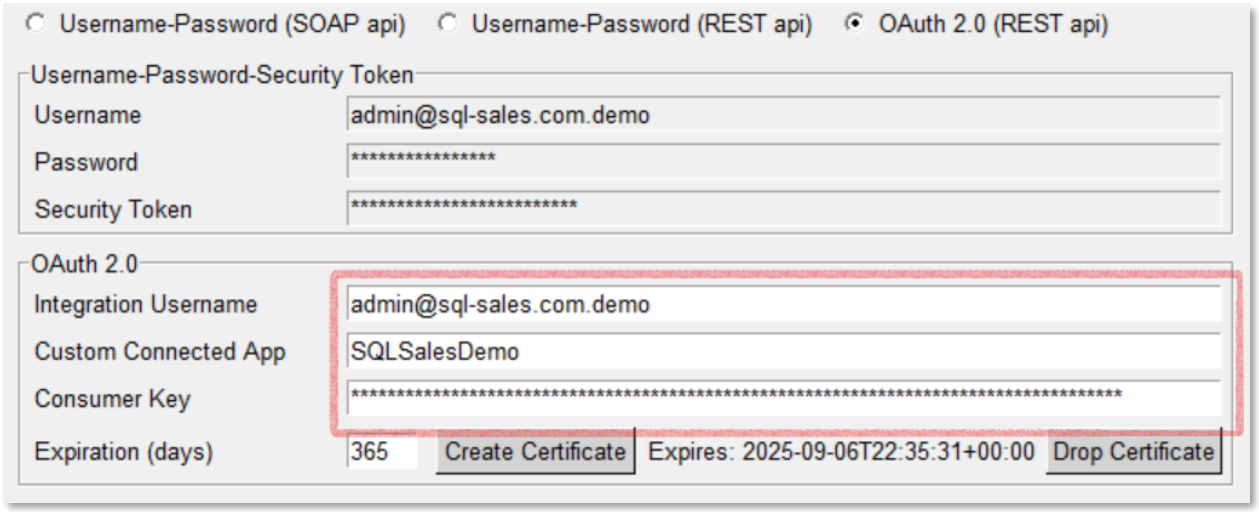


Step 17 - Copy the Consumer Key



Step 18 - In the SQL-Sales Environment configuration, select the "OAuth 2.0 (REST api)" connection setting and enter the following:

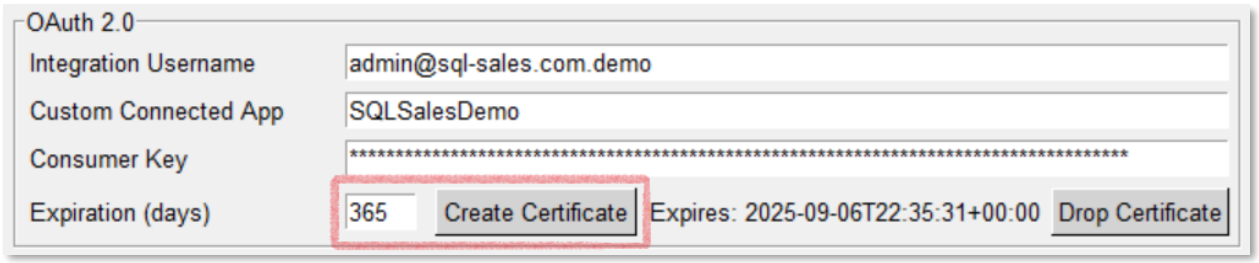
Input field	Notes
Integration Username	Enter the nominated username which will serve as the Integration User, this user's Profile must have been added in Steps 13-14 "Manage Profile"
Custom Connected App	This is the Name of your Connected App
Consumer Key	Paste here the copied Consumer Key from the prior step 17



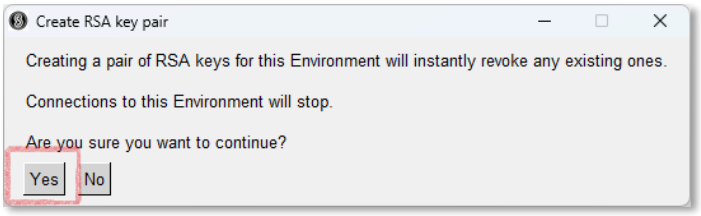
Step 19 - We'll now create the self signed certificate as mentioned in step 2.

Enter an expiry term in days (maximum is 365 days).

Click "Create Certificate"



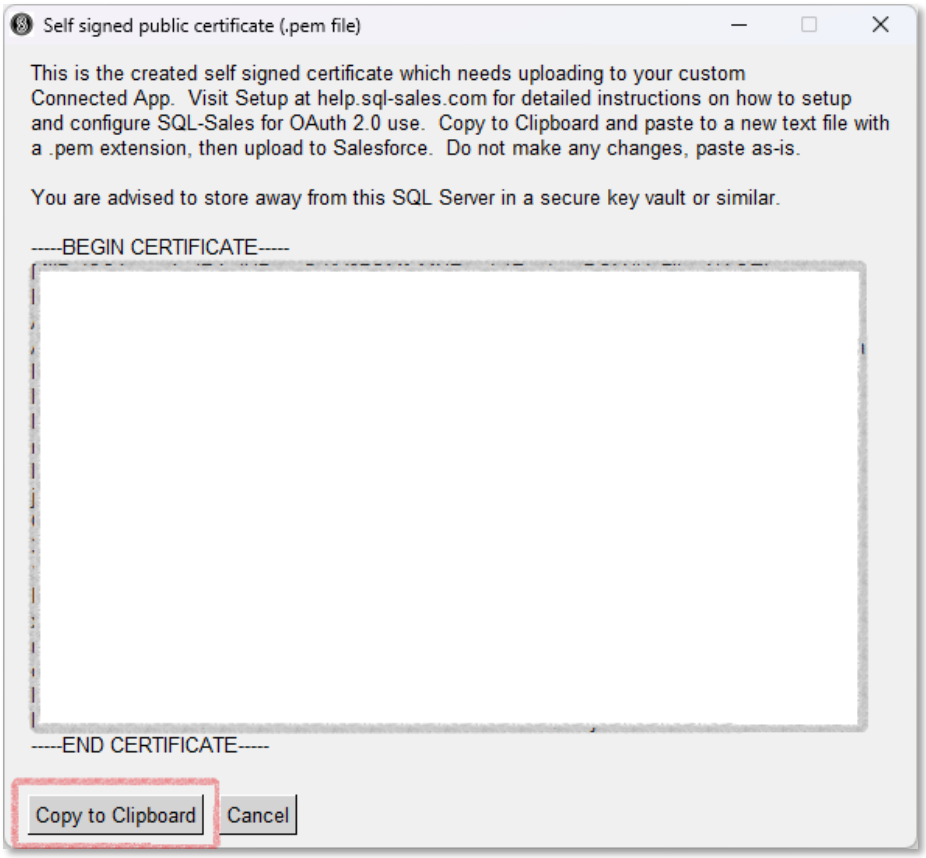
Click "Yes" at the confirmation prompt below:



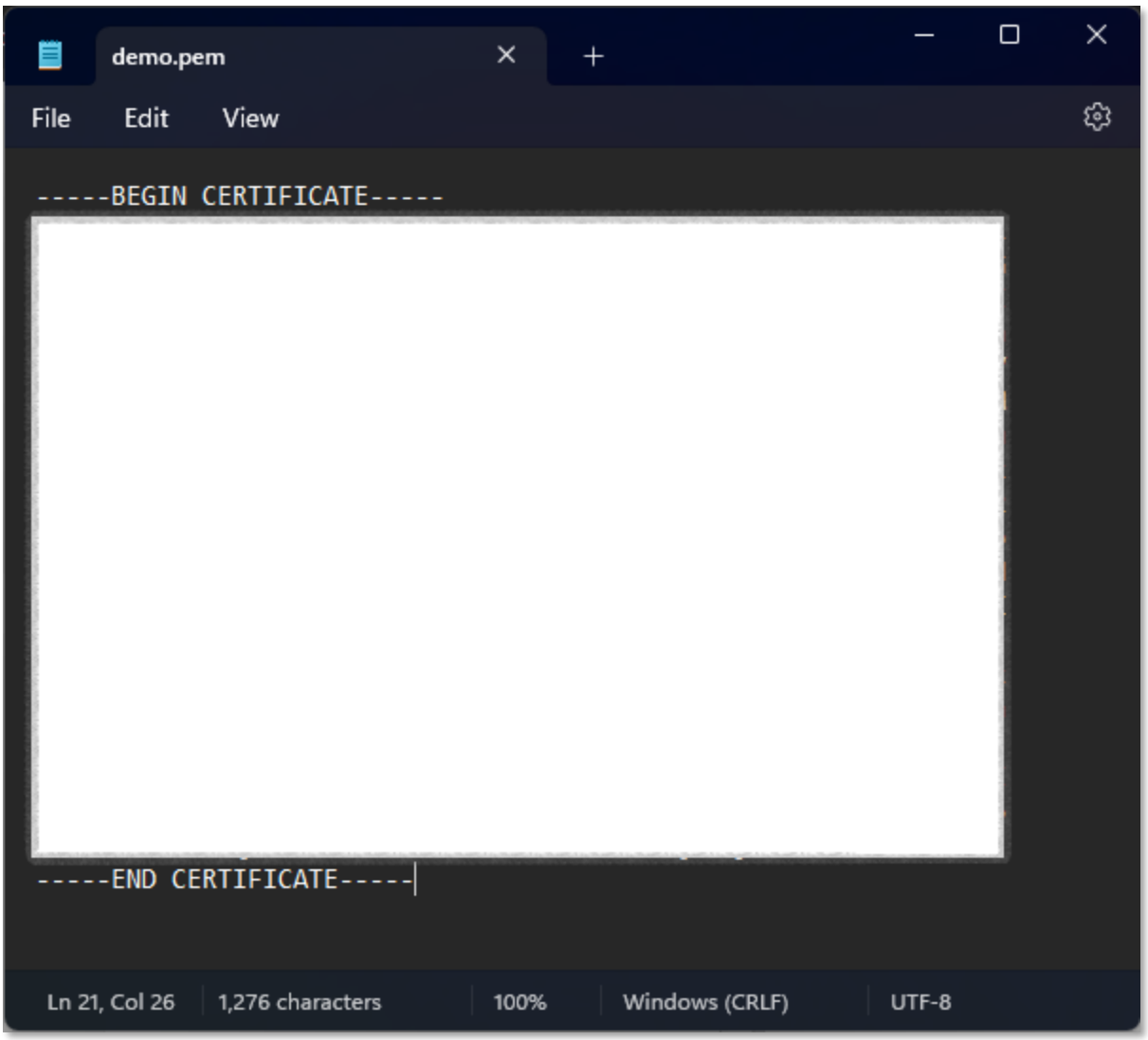
Step 20 - SQL-Sales will have generated a public self signed certificate for you to copy to your clipboard and save yourself as a .pem file to a location of your choosing. SQL-Sales will not hold or retain this beyond passing to the clipboard, as below.

Next steps:

1. Save as a suitably named file with a .pem extension
2. Save this to a key vault / safe location that you define and have control of
3. And will be able to browse to in the next section when you upload to Salesforce



Paste to a suitable text editor (for example notepad) and save as-is with no editing whatsoever.

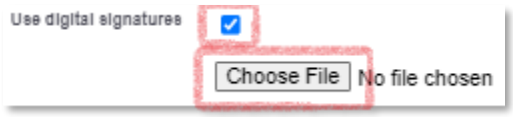


Step 21 - Edit your Connected App

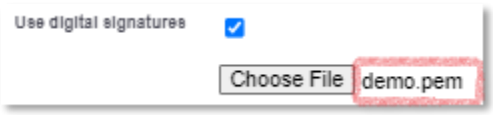
App Manager >> [Your Connected App] >> "Edit"

Tick "Use digital signatures"

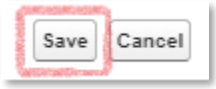
Click "Choose File"



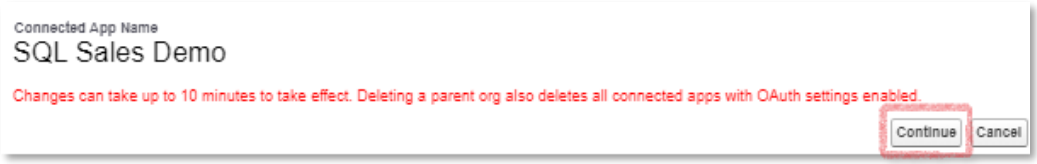
Browse to your .pem file, in the example here "demo.pem"



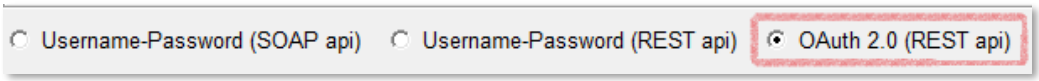
Click Save

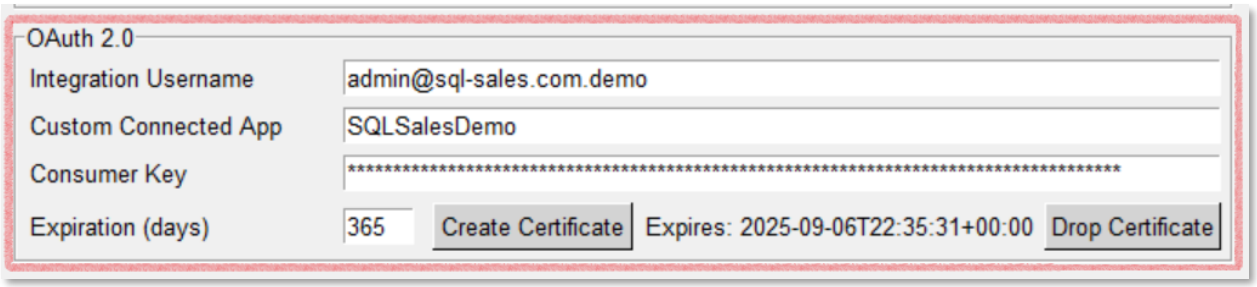


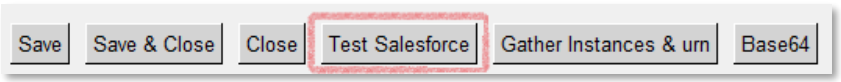
Salesforce informs there can be a delay of up to 10 minutes for the certificate to take effect, in reality this is typically instantly usable, click "Continue" at the prompt below:

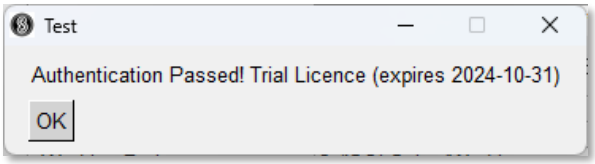


Step 22 - Finally, we can test in SQL-Sales



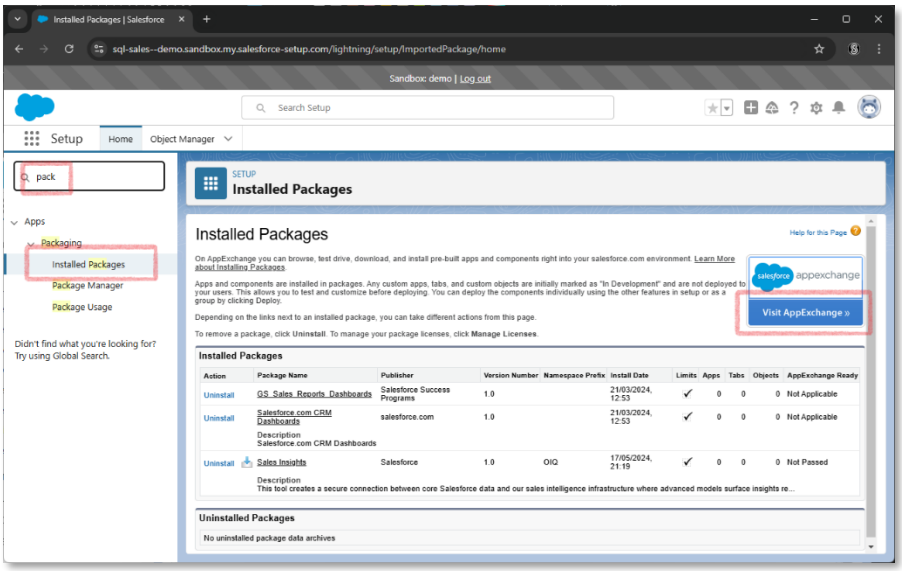


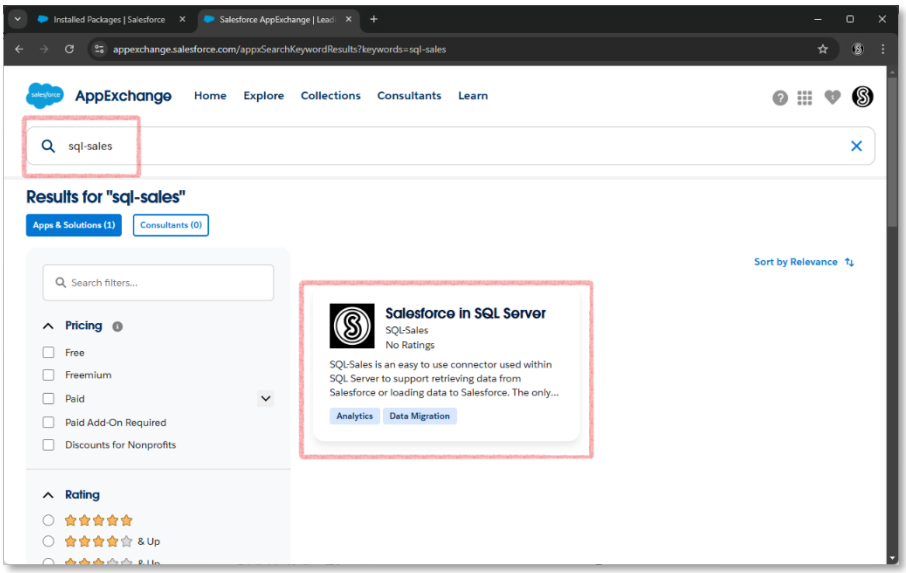




MANAGED PACKAGE INSTALLATION

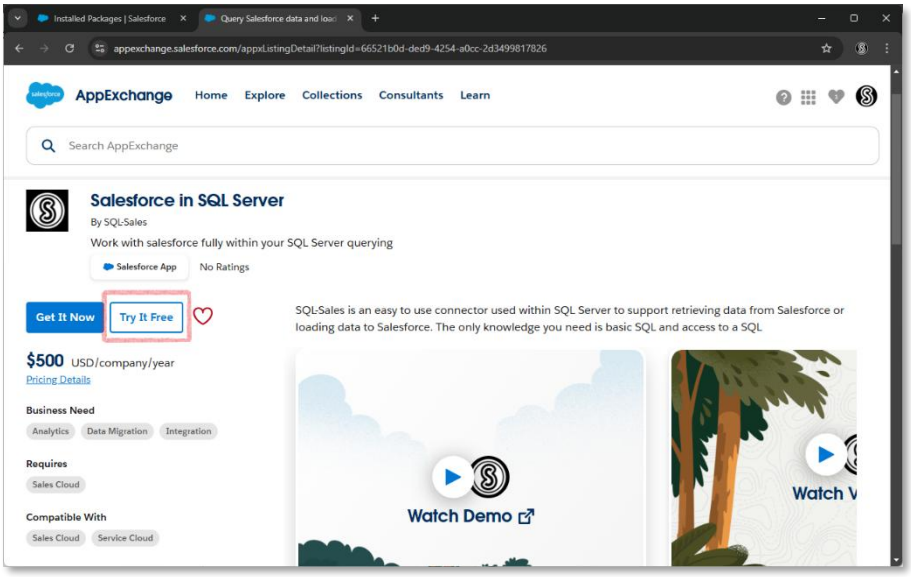
Install the Salesforce approved package from the Production or Sandbox by visiting the AppExchange. Search for Installed Packages and click on the "Visit AppExchange" button, as below:

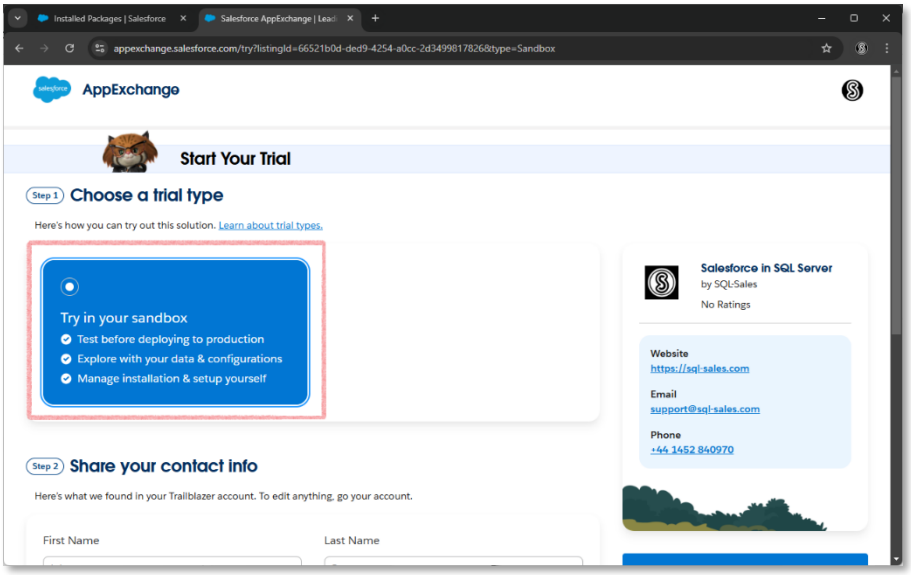


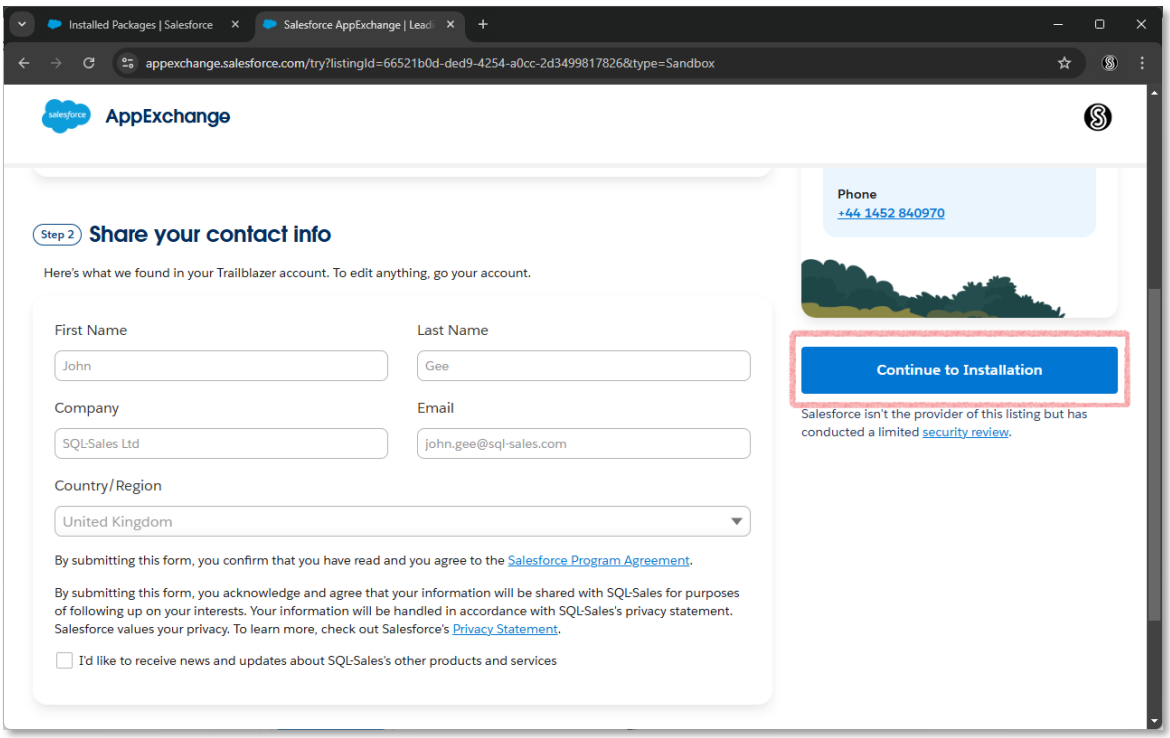


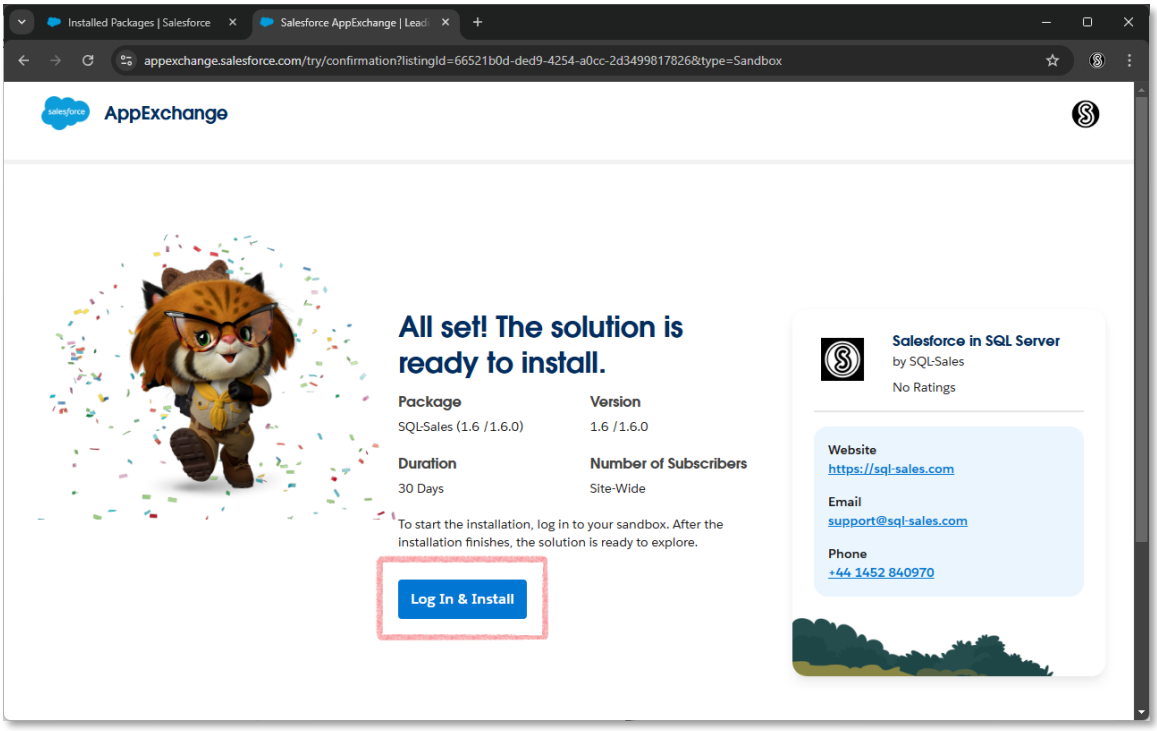
SQL-Sales

Note, “Try It Free” will only support a sandbox installation. Whereas “Get It Now” will install to your Production Org. Both methods are a free 30 day trial.

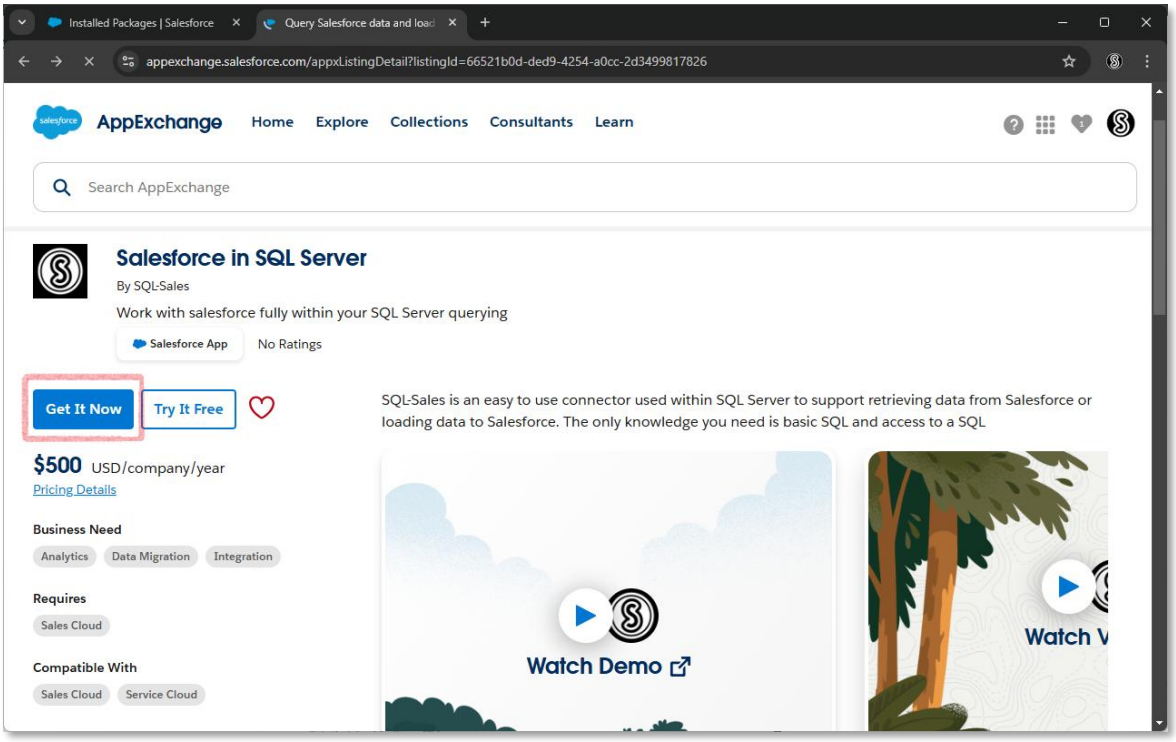


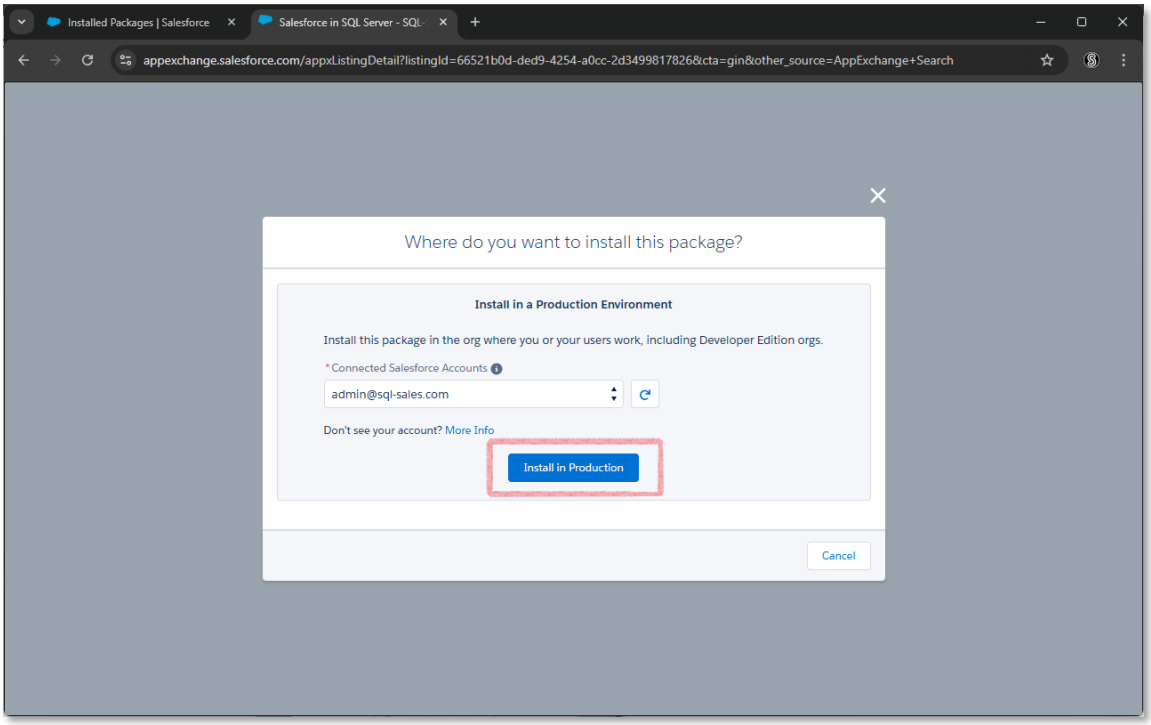



Installed Packages | Salesforce
Salesforce AppExchange | Lead
appexchange.salesforce.com/try?listingId=66521b0d-ded9-4254-a0cc-2d3499817826&type=Sandbox
AppExchange
Phone
+44 1452 840970
Step 2
Share your contact info
Here's what we found in your Trailblazer account. To edit anything, go your account.
First Name
John
Last Name
Gee
Company
SQL-Sales Ltd
Email
john.gee@sql-sales.com
Country/Region
United Kingdom
Continue to Installation
Salesforce isn't the provider of this listing but has conducted a limited security review.
By submitting this form, you confirm that you have read and you agree to the Salesforce Program Agreement.
By submitting this form, you acknowledge and agree that your information will be shared with SQL-Sales for purposes of following up on your interests. Your information will be handled in accordance with SQL-Sales's privacy statement. Salesforce values your privacy. To learn more, check out Salesforce's Privacy Statement.
I'd like to receive news and updates about SQL-Sales's other products and services

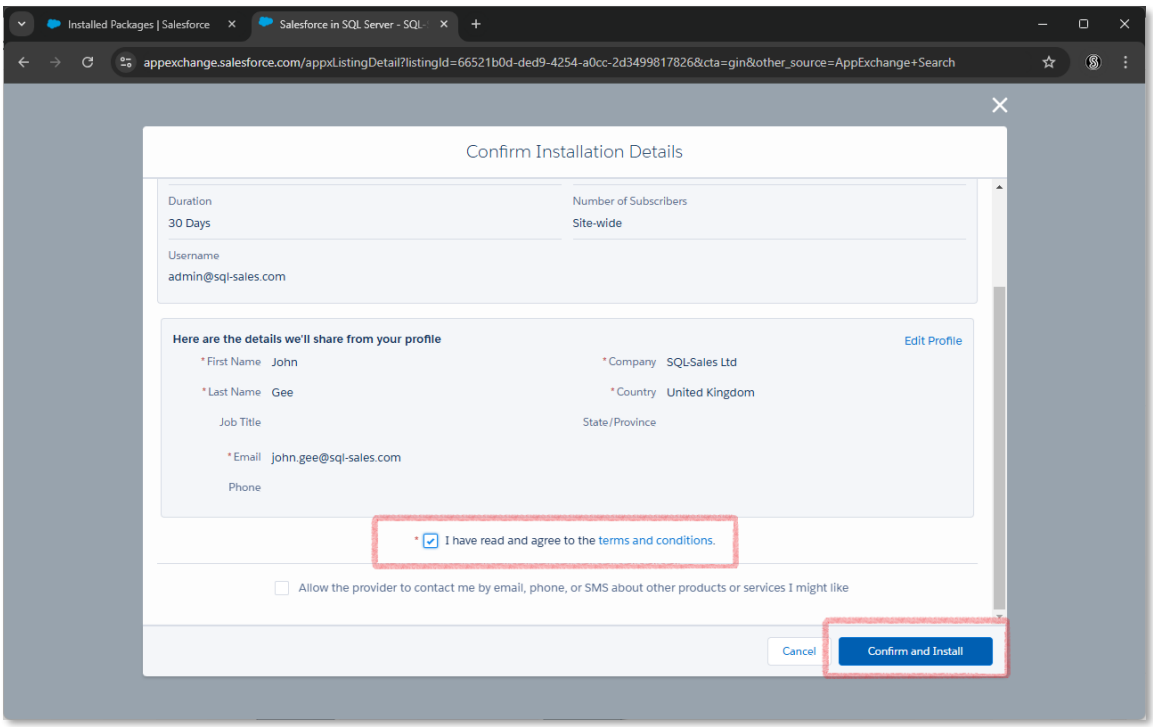

Installed Packages | Salesforce
Salesforce AppExchange | Lead
appexchange.salesforce.com/try/confirmation?listingId=66521b0d-ded9-4254-a0cc-2d3499817826&type=Sandbox
AppExchange
All set! The solution is ready to install.
Package
SQL-Sales (1.6 /1.6.0)
Version
1.6 /1.6.0
Duration
30 Days
Number of Subscribers
Site-Wide
To start the installation, log in to your sandbox. After the installation finishes, the solution is ready to explore.
Log In & Install
Salesforce in SQL Server
by SQL-Sales
No Ratings
Website
https://sql-sales.com
Email
support@sql-sales.com
Phone
+44 1452 840970

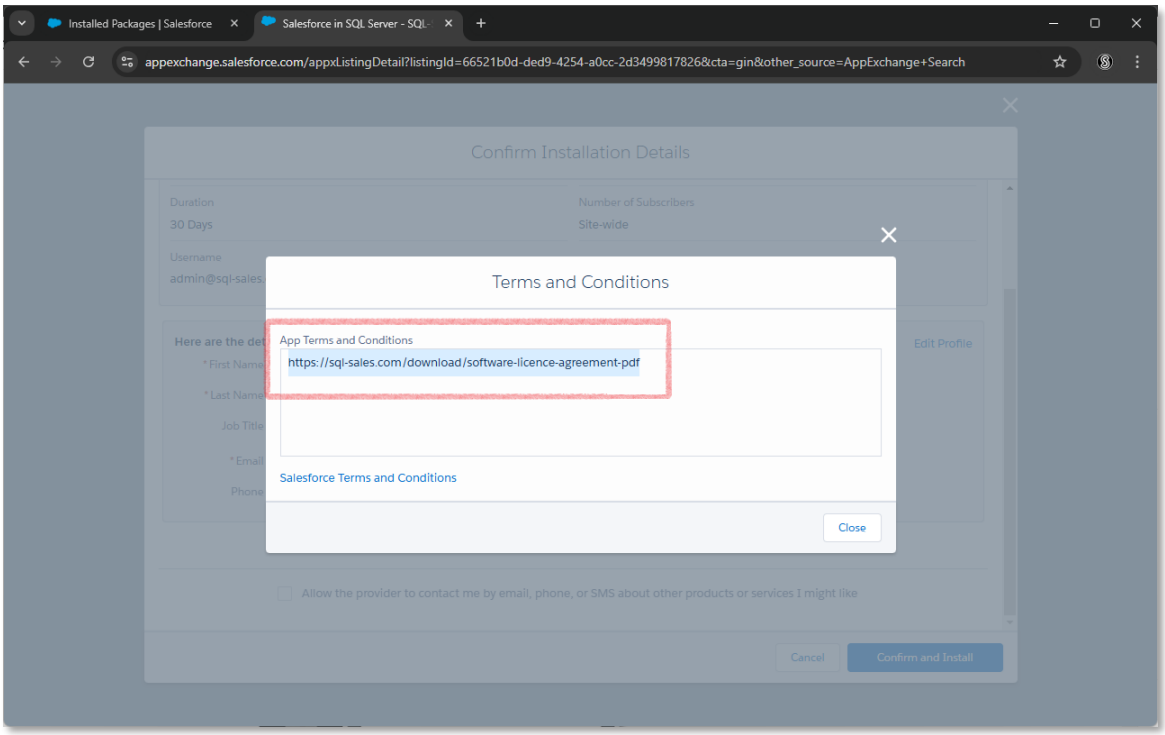
"Get It Now" for the Production install is also a 30 day free trial.



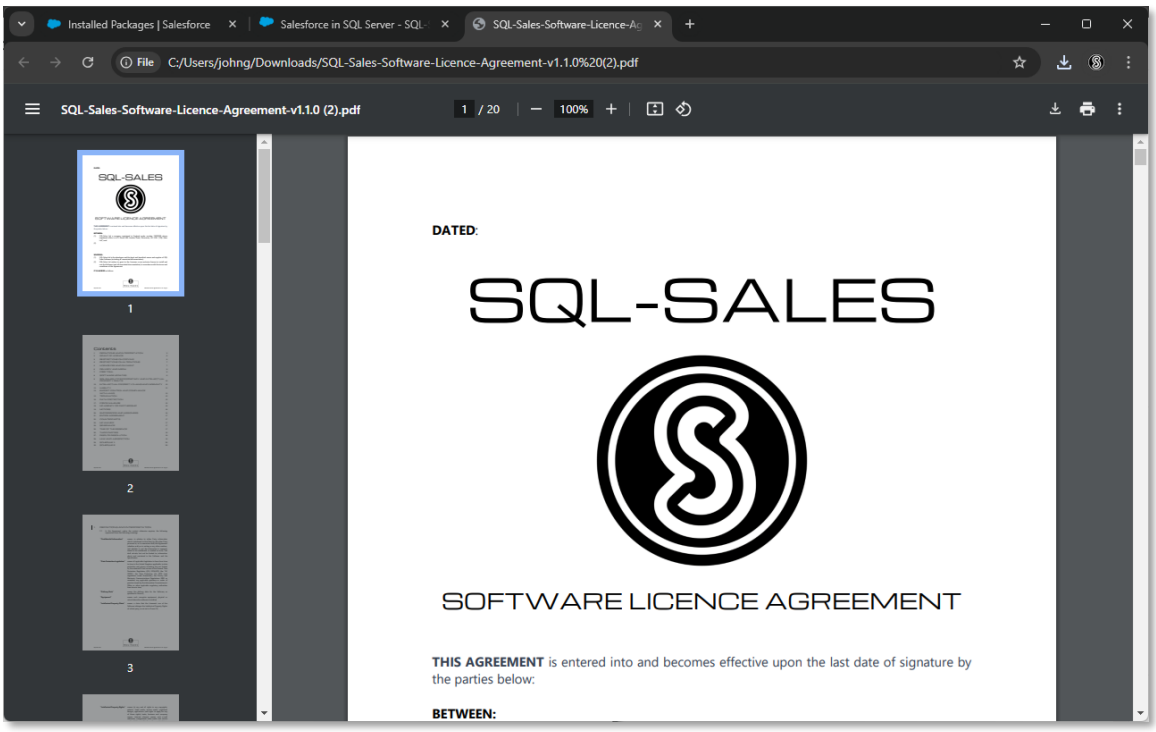


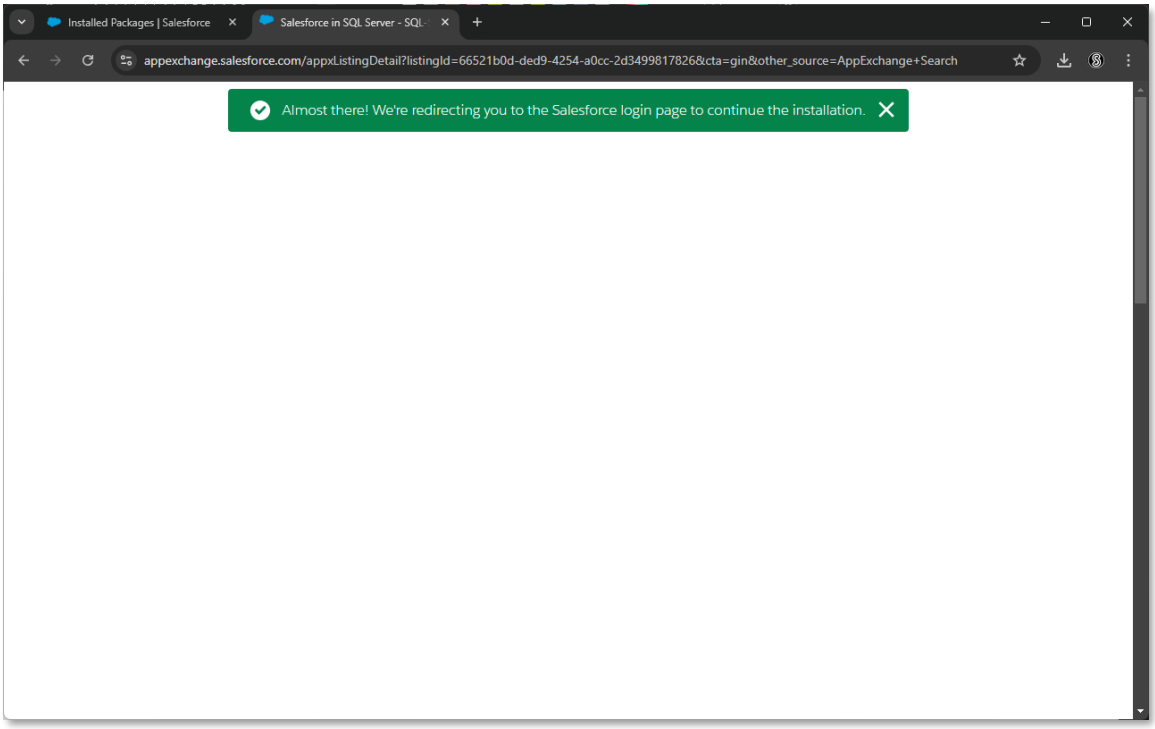
SQL-Sales


Installed Packages | Salesforce
Salesforce in SQL Server - SQL-
appexchange.salesforce.com/appxListingDetail?listingId=66521b0d-ded9-4254-a0cc-2d3499817826&cta=gin&other_source=AppExchange+Search
Confirm Installation Details
Duration
30 Days
Number of Subscribers
Site-wide
Username
admin@sql-sales.com
Here are the details we'll share from your profile
Edit Profile
*First Name John
*Last Name Gee
Job Title
*Email john.gee@sql-sales.com
Phone
*Company SQL-Sales Ltd
*Country United Kingdom
State/Province
I have read and agree to the terms and conditions.
Allow the provider to contact me by email, phone, or SMS about other products or services I might like
Cancel
Confirm and Install

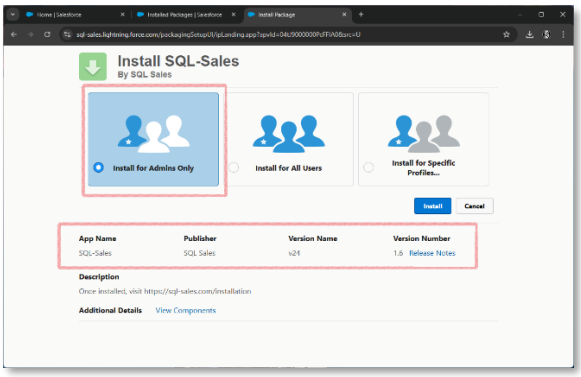

Installed Packages | Salesforce
Salesforce in SQL Server - SQL-
appexchange.salesforce.com/appxListingDetail?listingId=66521b0d-ded9-4254-a0cc-2d3499817826&cta=gin&other_source=AppExchange+Search
Terms and Conditions
App Terms and Conditions
https://sql-sales.com/download/software-licence-agreement-pdf
Salesforce Terms and Conditions
Close

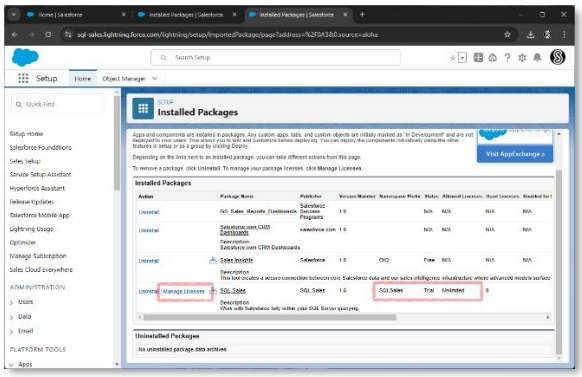
SQL-Sales


Installed Packages | Salesforce
Salesforce in SQL Server - SQL
SQL-Sales-Software-Licence-Ag
File C:/Users/johng/Downloads/SQL-Sales-Software-Licence-Agreement-v1.1.0%20(2).pdf
SQL-Sales-Software-Licence-Agreement-v1.1.0 (2).pdf
1 / 20
100%
DATED:
SQL-SALES
SOFTWARE LICENCE AGREEMENT
THIS AGREEMENT is entered into and becomes effective upon the last date of signature by the parties below:
BETWEEN:


Installed Packages | Salesforce
Salesforce in SQL Server - SQL
appexchange.salesforce.com/appxListingDetail?listingId=66521b0d-ded9-4254-a0cc-2d3499817826&cta=gin&other_source=AppExchange+Search
Almost there! We're redirecting you to the Salesforce login page to continue the installation.

"Get It Now" for the Production install is still a 30 day free trial. As below, the package will be installed with a Status = "Trial". If you click in to "Manage Licenses" you'll see the Expiration Date.

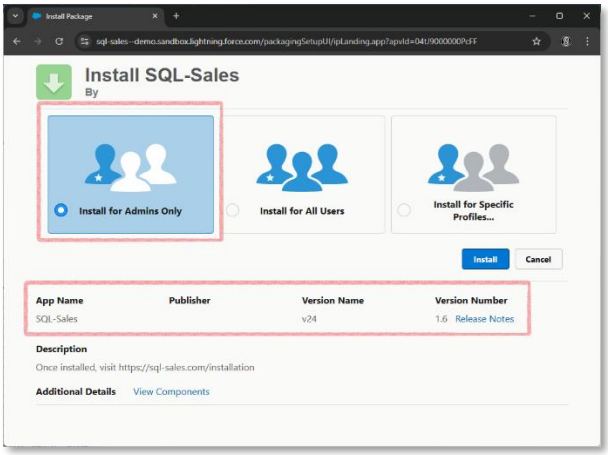




Note, once the package is installed, you can immediately start using SQL-Sales by connecting in the Configuration tool with Username-Password (SOAP or REST api). However if you wish to connect with OAuth 2.0 (REST api) you will need to make the configuration changes in the steps that follow.

Installing the Package

Choose "Install for Admins Only"

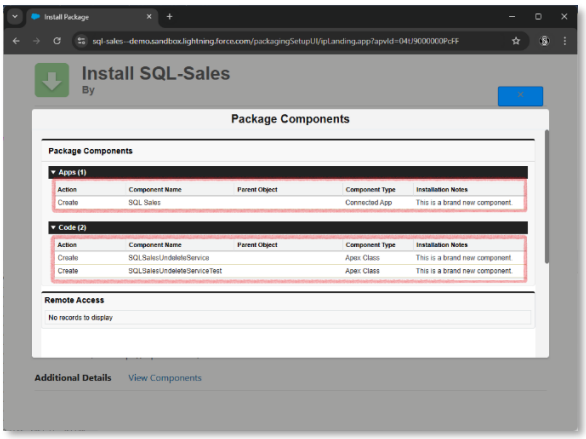


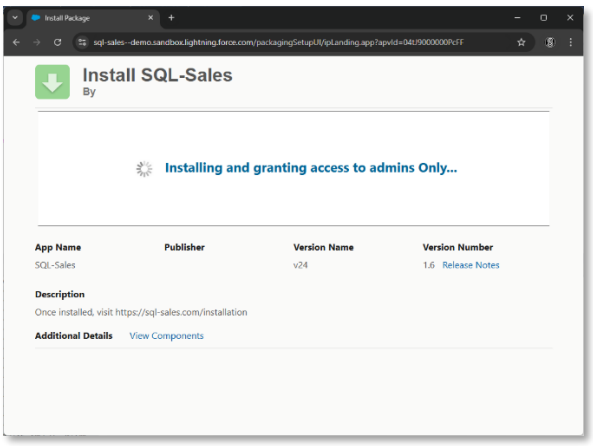
"View components"

The "SQLSalesUndeleteService" Apex Class (and associated Test Class) provides a custom REST api, supporting undelete functionality. This allows a full range of data operations via the OAuth 2.0

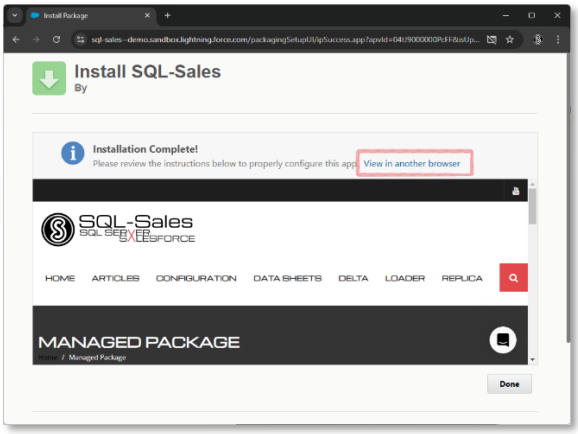
connection (as undelete is not supported by default by the Salesforce REST api, only the SOAP api supports this by default).

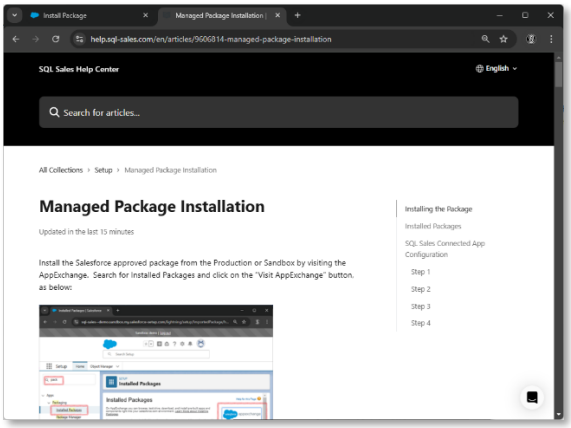
The "SQL Sales" Connected App handles the connection itself (from the installed SQL-Sales Daemon).



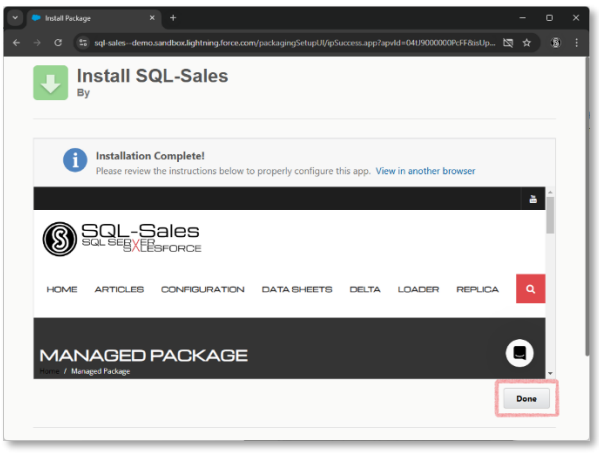


One you get to “Installation Complete!” - if you click on "View on another browser", you’ll be directed to the Managed Package help pages (the same information you’re reading here) previewed in the frame below.

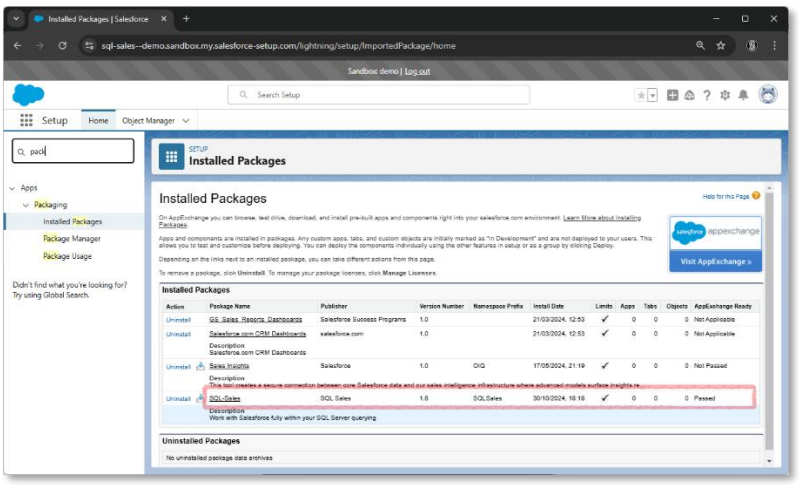




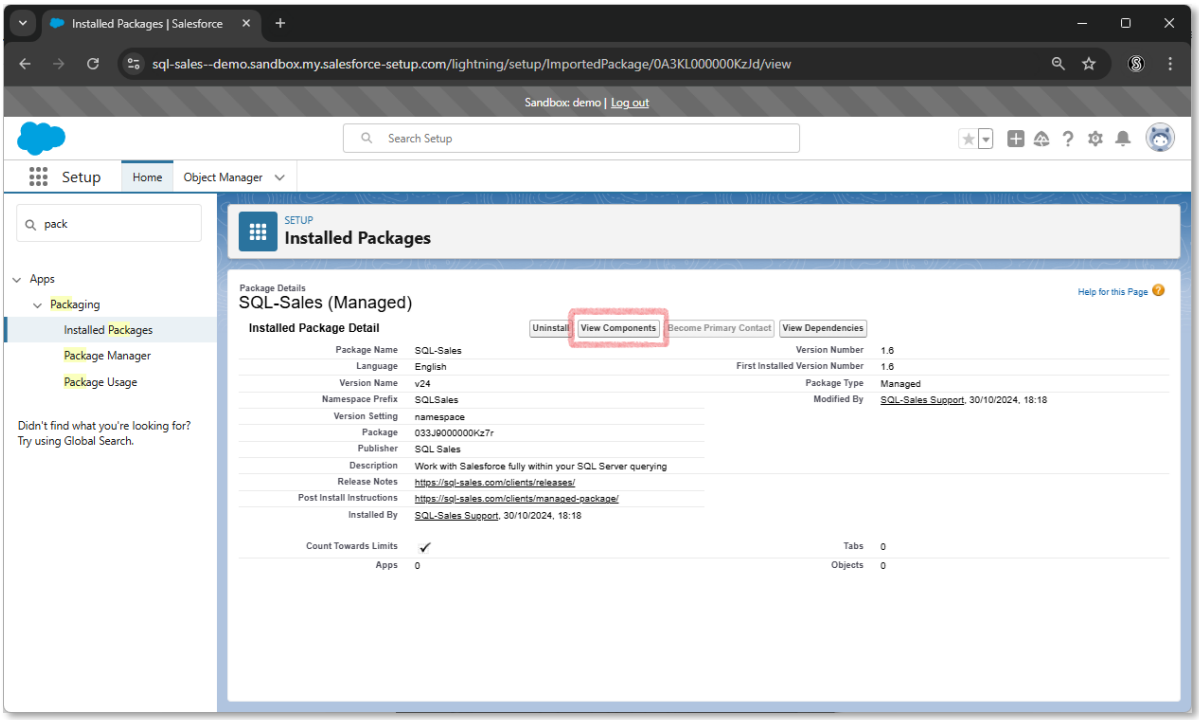
Finally Click "Done"



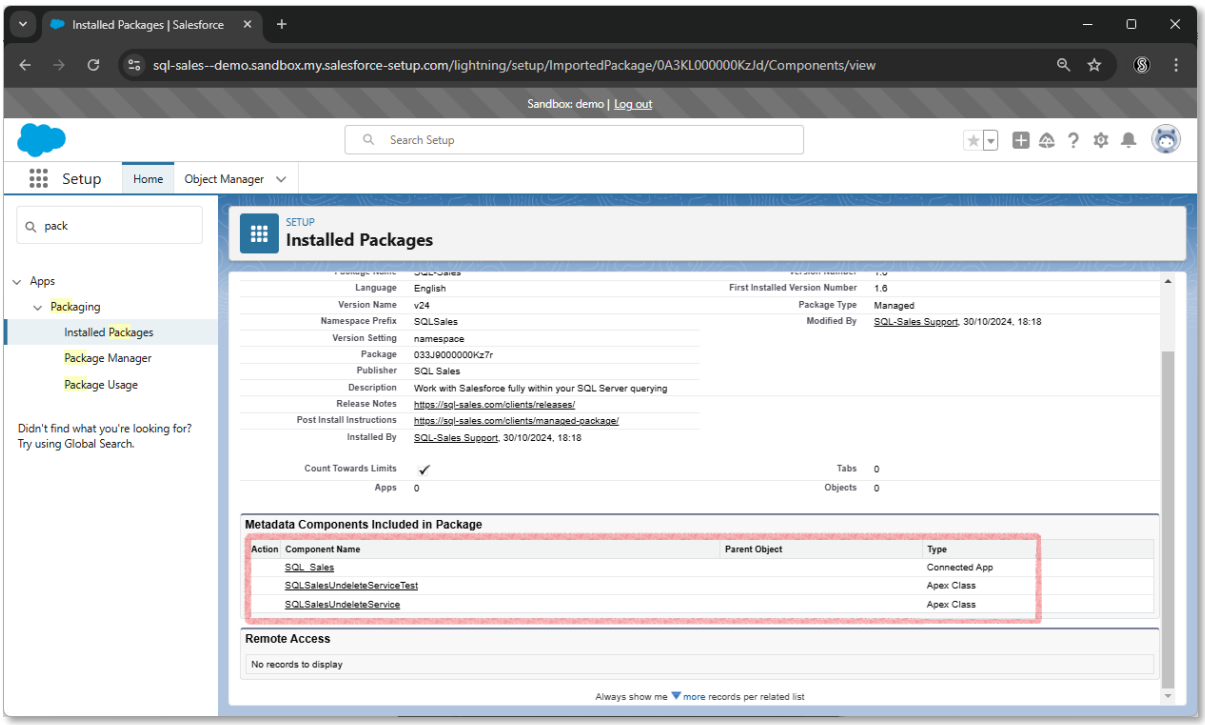
Installed Packages - click onto "SQL-Sales"



View Components: Click "View Components" to see what you have installed



As previously described - the package consists of a Connected App and an Apex class.



SQL Sales Connected App Configuration

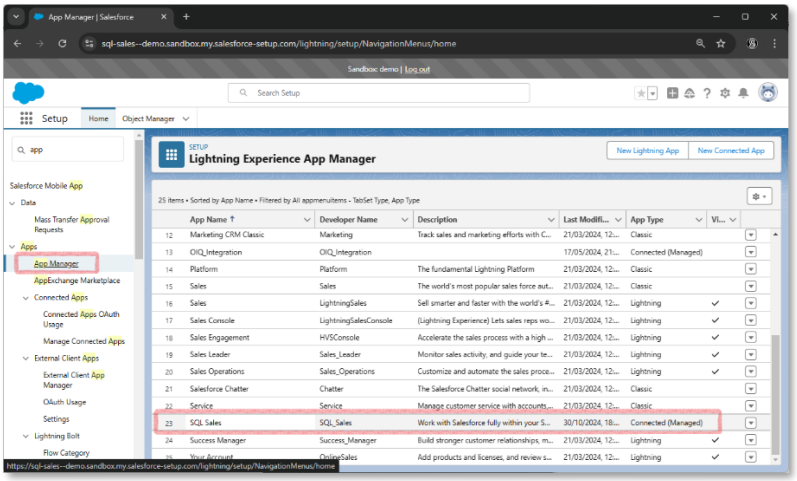
Note, once the package is installed, you can immediately start using SQL-Sales by connecting in the Configuration tool with Username-Password (SOAP or REST api).

However if you wish to connect with OAuth 2.0 (REST api) you will need to make some post installation configuration changes.

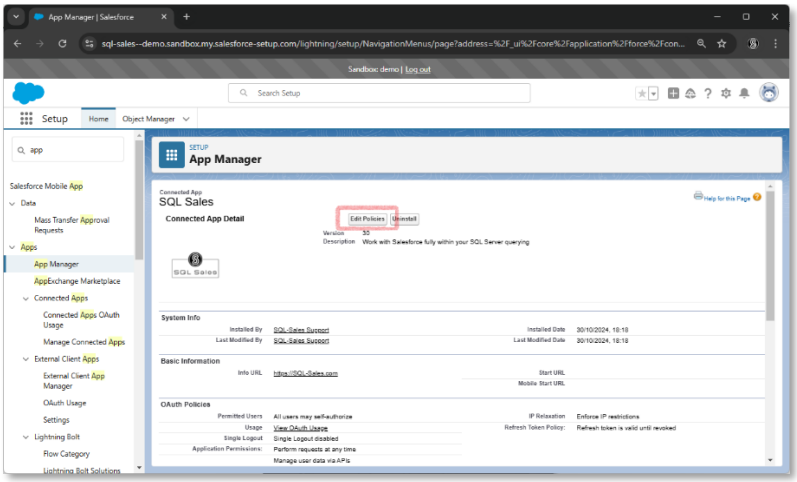
From Setup, type "app" and navigate to Apps >> App Manager.

You will have an entry for "SQL Sales" Connected (Managed).

Click "Manage"



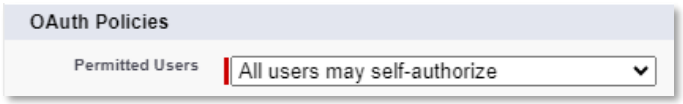
Click "Edit Policies"



OAuth Policies

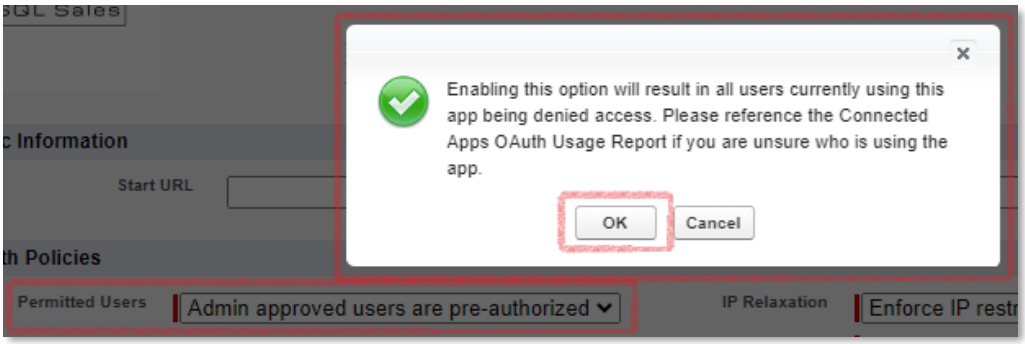
Step 1

By default your installed connected app will have Permitted Users set to: "All users may self-authorize"



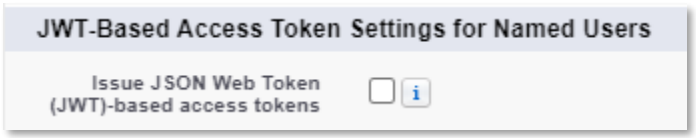
Change Permitted Users to: "**Admin approved users are pre-authorized**"

Click OK on the confirmation prompt below:

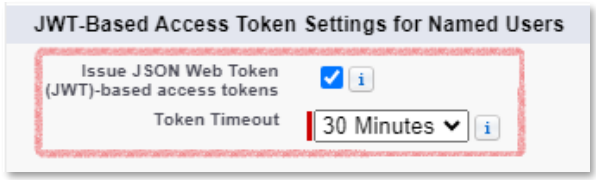


Step 2

By default "Issue JSON Web token (JWT)-based access tokens is not enabled



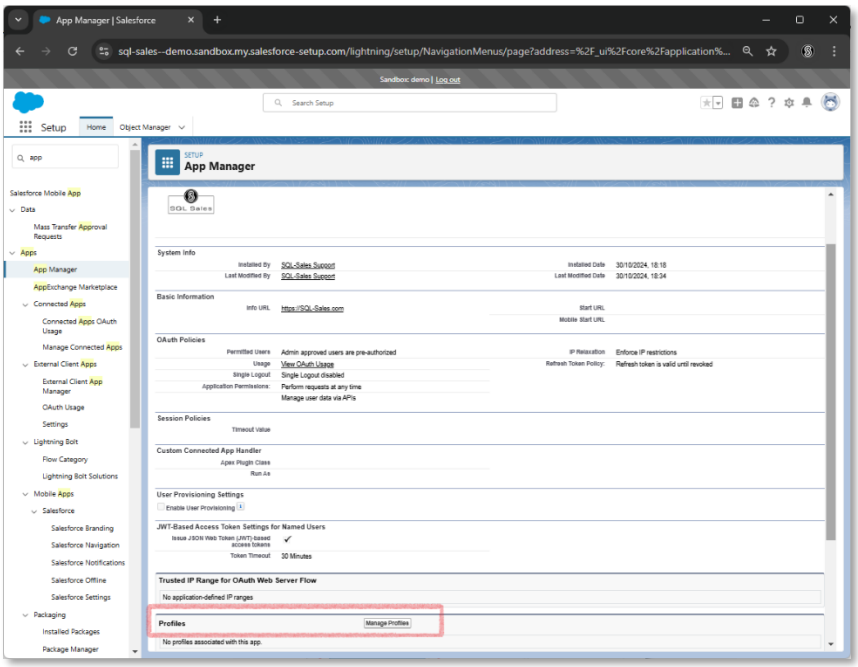
Tick the checkbox and leave the default token timeout at 30 Minutes (SQL-Sales at run time will validate this setting is 30 minutes).



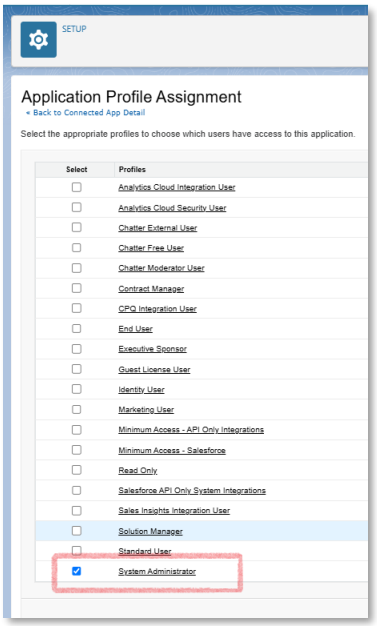
Click Save

Step 3

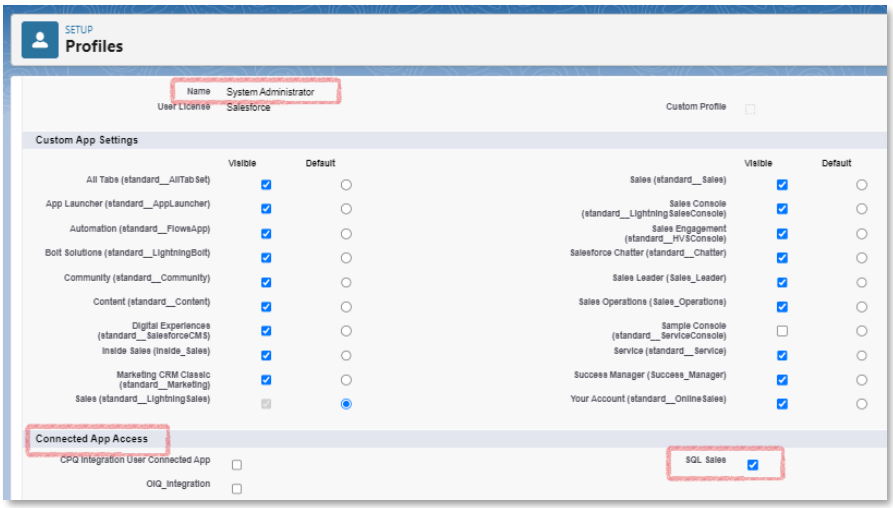
Back on the SQL Sales Connected App Detail page, click on "Manage Profiles"



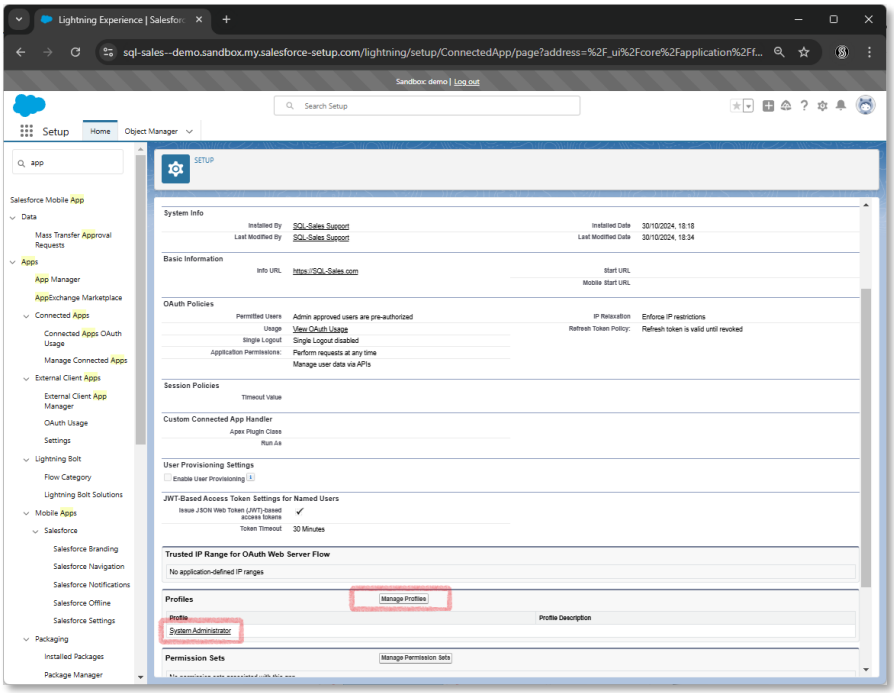
Click "System Administrator"



For completeness and general awareness, you can tick the "SQL Sales" checkbox in the Connected App Access section of the System Administrator Profile page.



Either method will result in the below, on the SQL Sales Connected App detail page:



Step 4

Finally, you **must** now follow the instructions here to setup your connection for the given Salesforce instance you wish to connect to (in our example here, the "demo" sandbox)

SQL-Sales

DATABASE ENABLING

ss_EnableDatabase.sql

Provided alongside the SQLSalesInstaller.exe, the "enabling" script ss_EnableDatabase.sql will fully deploy the required SQL Server components (stored procedures and functions).

ss_EnableDatabase.sql is installed to the exe install location - typically to the default C:\Program Files\SQLSales, hence: C:\Program Files\SQLSales\ss_EnableDatabase.sql

Open and compile to your chosen database, to "enable" that database for use with SQL Sales.

Note, SQL Sales requires that xp_cmdshell is enabled on your SQL Server. If it is not, the enabling script will identify that and provide a suggested script to run, to alter your system configuration settings.

The following are deployed to your target (schema) and database by running the enabling script, in the cases of some tables such as ss_Log, these will be auto-deployed on first running certain stored procedures:

Object	Type	Purpose
ss_Admin	Stored Procedure	Primarily to control the Daemon from your SSMS client
ss_Delta	Stored Procedure	Delta Replication of Salesforce data
ss_DeltaAll	Stored Procedure	Call ss_Delta in bulk for multiple Objects
ss_Handler	Stored Procedure	Generic process used by most stored procedure calls to interact with the Handler application
ss_Loader	Stored Procedure	Allows bulk loads to Salesforce via the SOAP and Bulk APIs (Version 1 & 2)
ss_LoaderChecks	Stored Procedure	Supports ss_Loader
ss_Logger	Stored Procedure	Controls the logging to ss_Log_Working
ss_MetaField	Stored Procedure	Returns Field metadata
ss_MetaObject	Stored Procedure	Returns Object metadata
ss_MetaPick	Stored Procedure	Returns Picklist metadata
Ss_ObjectLog	Stored Procedure	Controls the logging to ss_Log
ss_PullChecks	Stored Procedure	Supports ss_Replica & ss_Delta
ss_Replica	Stored Procedure	Full Replication of Salesforce data
ss_ReplicaAll	Stored Procedure	Call ss_Replica in bulk for multiple Objects
ss_UserInfo	Stored Procedure	Miscellaneous user information for a given Environment
ss_AutoReplica	Table	Controls if a given Object is automatically fully replicated when calling ss_DeltaAll
ss_BulkAPILog	Table	Log table used by ss_Loader and the Bulk API

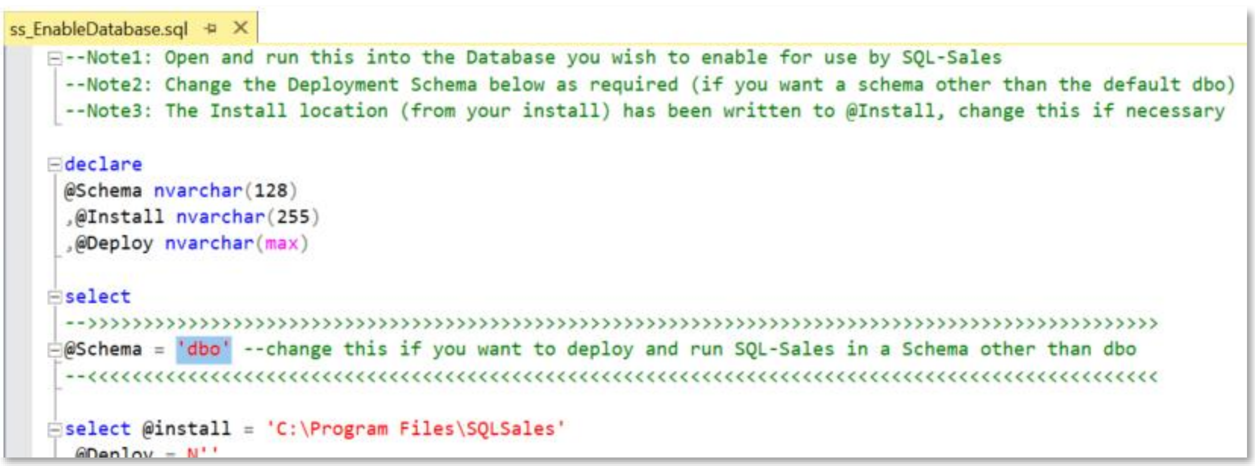
Object	Type	Purpose
ss_Log	Table	Log table for all SQL Sales Delta and Full replications for a given Salesforce object
ss_Log_Working	Table	Finer logging detail to ss_Log
ss_ObjectExclusion	Table	Control which objects are excluded from ss_ReplicaAll and ss_DeltaAll
ss_ObjectInclusion	Table	Control which objects are included from ss_ReplicaAll and ss_DeltaAll
ss_SysField	Table	Output from ss_MetaField
ss_SysObject	Table	Output from ss_MetaObject
ss_SysPicklist	Table	Output from ss_MetaPicklist
ss_System	Table	Contains the deployed SQL Sales Version
ss_18	Function	Helper function to generate the full 18 character case insensitive Salesforce Id from a 15 character case sensitive Id.
ss_LoaderDefinition	Function	Supports ss_Loader
ss_SchemaDefinition	Function	Supports ss_Replica & ss_Delta
ss_System	Table	Contains the deployed SQL Sales Version and general future use
ss_Log	Table	Log table for all SQL-Sales Delta and Full replications for a given Salesforce object (will only be auto-generated on the first ss_Replica run)
ss_Log_Working	Table	Log table for all stored procedures. Self-maintained (any entries > 7 days old are auto-purged)
ss_BulkAPILog	Table	Log table used by ss_Loader and the Bulk API (will only be auto-generated on the first ss_Loader run involving the Bulk API)
ss_ObjectExclusion	Table	Referenced by ss_ReplicaAll and ss_DeltaAll, add any objects into here to have them excluded
ss_ObjectInclusion	Table	Referenced by ss_ReplicaAll and ss_DeltaAll, add any objects into here to form an inclusion-only set of objects (any object not in here will be excluded from scope)

Full Schema isolation support

SQL Sales can run in separate schema on the same databases, which can be helpful for certain use cases or where for example you can only operate in one database and need to have different sandboxes replicated to different schemas, even if necessary having Production replication in their own "prod" schema (although generally speaking as best practise we would advise to physically separate a Production replication to its own dedicated database and ideally to its own SQL Server), that said we recognise some customers have to operate within very restricted SQL environments, which is why full schema isolation can be a real benefit.

Setup

When enabling a given database, simply change the highlighted code block, specifying your required schema (unless the default dbo schema is required in which case do nothing). Now execute the script.



Working with schemas

As with standard SQL Server, you do not need to specify the dbo schema when calling stored procedures or selecting from tables, if that is the schema you specified on your given installation (which is the usual default), hence using ss_Replica as an example:

```
exec ss_Replica 'DEMO', 'Account'
select * from Account
```

you can, if you prefer also run as

```
exec dbo.ss_Replica 'DEMO', 'Account'
select * from dbo.Account
```

Let's say you have enabled with schema "uat" (for a User Acceptance Testing sandbox) and a second schema called "dev"(for a Development sandbox), you can run the below:

```
exec uat.ss_Replica 'UAT_SBOX', 'Account'
exec dev.ss_Replica 'DEV_SBOX', 'Account'
```

and thereby create physically separate replicas in tables:

uat.Account

dev.Account

```
select * from uat.Account
select * from dev.Account
```

FULL REPLICATION

ss_Replica

ss_Replica is a Stored Procedure you execute within your enabled database.

Prerequisites

- A working Environment and the SQL Sales Daemon is running (see Environment Setup)
- SQL Server
- Compiled stored procedure ss_Replica (see Database Enabling)

ss_Log replication log

Replication is logged in table "ss_Log" which is automatically created and maintained for all ss_Replica runs and the same for ss_Delta runs (see next section).

Note, the table is created within the schema to which you are running ss_Replica, hence if you ran with uat.ss_Replica, the resultant entry for the given object will be to table uat.ss_Log.

Field	Purpose
ReplicaName	The Salesforce object name or a Custom table name if one has been specified, created as a replica table in the given enabled Database
ObjectName	Source Salesforce Object of this replication log (will be the same as ReplicaName if no customisation has occurred with the "Table:" switch
CustomReplica	Indicates if the Replica is Custom (i.e. not a basic replication of a Salesforce Object Name but ReplicaName is a custom table name
TableCreatedDate	Serves no functional purpose, is for information purposes only
MaxSystemDate	This is how the ss_Delta process determines how to delta replicate
LogDate	Last log datetime
TopRow	Captures any Top commands, for example 'Account:Top100'
Subset	Captures any Column Subset commands, for example 'Subset(Name,BillingCountry)'
Status	Possible values: FAILURE (indicates a failure has occurred, see Detail field) FULL (indicates a Full replication occurred) DELTA (indicates a partial replication occurred, via ss_Delta)
WhereInput	Captures any Where clause commands injected into the replication
Detail	Success or Failure detail

ss_Log_Working detailed replication log

For more granular detail of every replication run, each step is logged in "ss_Log_Working" which can be helpful in monitoring longer running processes of millions of rows, as the BULK connection method will deliver the data payload in chunks, which are logged and so can be monitored to check progress.

Field	Purpose
Environment	The specified Salesforce Environment, defined in the SQL Sales Configuration tool
Task	"ss_Replica"
ObjectName	Source Salesforce Object of this replication log (will be the same as ReplicaName if no customisation has occurred with the "Table:" switch
PreProcessPoint	The process which is about to be started
Detail1	The Salesforce object name or a Custom table name if one has been specified, created as a replica table in the given enabled Database
Detail2	The contents of parameter @Special1
Detail3	The contents of parameter @Special2
Detail4	Logs the current Chunk, if a Salesforce Id, this is the latest Id the process has got to

Parameters

Parameter	Purpose
@Env	SQL Sales Environment Name
@ObjectName	Salesforce Object Name (for example Account) Can be accompanied with :**Top**XXX where XXX is a number exec ss_Replica 'DEMO', 'Account:Top10' Can have an "**_All**" suffix to return soft deleted and/or archived data exec ss_Replica 'DEMO', 'Account_All'
@Special1	Optional, in simple usage, null **Subset(field1,field2,field3)** – where field1 etc are Salesforce api fieldnames for the given object. For example in the case of the object being Account, this could be: Subset(Name,Phone,BillingCity) **Table:<custom table name>** - for example 'Table:OppTest' No special delimiter is required between Subset(x,y,z) and Table:xxx, for example leave nothing or a space:

Parameter	Purpose
	'Subset(Name,Phone,BillingCity) Table:AccountTest' **LinkedEntityId** Only relevant if you are replicating Object 'ContentDocumentLink' see the relevant detail in this section, for example Exec ss_Replica 'demo', 'ContentDocumentLink','LinkedEntityId','a,b,c,d' (where 'a,b,c,d' is a comma delimited list of up to 200 Entity Ids for example Account Ids). **ContentDocumentId** Only relevant if you are replicating Object 'ContentDocumentLink' see the relevant detail in this section, for example Exec ss_Replica 'demo', 'ContentDocumentLink','ContentDocumentId','a,b,c,d' (where 'a,b,c,d' is a comma delimited list of up to 200 ContentDocument Ids)
@Special2	Optional, in simple usage, null **Where <SOQL>** - for example 'Where StageName = ''Closed Won''' If @Special1 = LinkedEntityId Provide a comma delimited string of up to 200 Entity Id (for example 200 Account Ids) If @Special1 = ContentDocumentId Provide a comma delimited string of up to 200 ContentDocumentIds

Simple Example

```
exec ss_Replica 'DEMO', 'Account'
```

Specifying Batchsize

By default ss_Replica will use a batchsize of 2000 (the maximum possible via the Salesforce SOAP api). Should you wish to use a smaller batchsize, specify this by passing in the custom batchsize when passing in the Objectname as below:

```
exec ss_Replica 'DEMO', 'Account(25)'
```

Retrieving soft deleted & archived data (Query All)

By default ss_Replica will automatically exclude IsDeleted = 1/True records as well as isArchived = 1/True. To include these records, append "_All" to your specified Objectname as below, note ss_Delta does not support the "_All" switch as it is not possible to reliably validate if a local IsDeleted = 1/True has been auto-purged from the Salesforce recyclebin.

```
exec ss_Replica 'DEMO', 'Account_All'
```

Quick Top check

This doesn't replicate back to the given object table name as it by default outputs as a resultset to the management studio results pane, to facilitate quick checks of data and metadata. However the same data is also present in a <ObjectName> + _Check table, for example Account_Check. Note (see following documentation) – combining Top with a custom replica table will result in the Top number of output rows being written to the specified SQL replica table as opposed to it being an instruction for the output to the screen only. Output to screen:

```
exec ss_Replica 'DEMO', 'Account:Top5'
```

Output to the specified custom replica table:

```
exec ss_Replica 'DEMO', 'Account:Top5','Table:AccountTopTest'
```

Subset Example

```
exec ss_Replica 'DEMO', 'Account','Subset(Name,BillingCountry)'
```

Valid fields (i.e. that exist on the specified object and for which you have permission to query, will only be returned, although note some basic fields to support the replication logic to function correctly will be imposed (for example Id, isDeleted; SystemModStamp).

Note, for larger sets of subset fields, you may prefer to input in a list form, example shown below, which is an acceptable input method to the @Special1 input parameter:

```
exec ss_Replica 'DEMO', 'Opportunity'
,'Subset(
AccountId
,Amount
,CampaignId
,CloseDate
,Name
,NextStep
,OrderNumber__c
,OwnerId
,Pricebook2Id
,Probability
,StageName
,SystemModstamp
,TotalOpportunityQuantity
,TrackingNumber__c
,Type)'
```

Or

```
exec ss_Replica 'DEMO', 'Opportunity'
,'Subset(
AccountId,
Amount,
CampaignId,
CloseDate,
Name,
NextStep,
OrderNumber__c,
OwnerId,
Pricebook2Id,
Probability,
StageName,
SystemModstamp,
TotalOpportunityQuantity,
TrackingNumber__c,
Type)'
```

Combined Subset, Batchsize and Top check example

```
exec ss_Replica 'DEMO', 'Account(25):Top100','Subset(Name,BillingCountry)'
```

Custom Replica Example

```
exec ss_Replica 'DEMO', 'Opportunity','Table:OppTest'
```

In this example, Salesforce Opportunity data will be replicated to the "OppTest" SQL table. This can be maintained with ss_Delta if required, entirely independently of any potential "Opportunity" replica

```
exec ss_Delta 'DEMO', 'Opportunity', null, 'Table:OppTest'
```

existing in the same deployed schema in your given Database.

For example running

```
exec ss_Replica 'DEMO', 'Opportunity','Table:OppTest'
exec ss_Replica 'DEMO', 'Opportunity'
```

will define two separate replica tables of Salesforce Opportunity data, one called Opportunity, the other called OppTest, which can both be maintained via ss_Delta, provided the "Table:" switch is applied consistently:

```
exec ss_Delta 'DEMO', 'Opportunity', null, 'Table:OppTest'
exec ss_Delta 'DEMO', 'Opportunity'
```

Combining Top with Custom Replica

If Top is combined with a custom replica instruction, the limit of rows is applied to the custom created replica object and therefore there is no _check table output to the SSMS results pane.

Where Clause Example

```
exec ss_Replica 'DEMO', 'Opportunity','Table:OppTest','Where StageName = ''Closed Won'''
```

In this example, a where clause has been added, which will be applied to the replica and can be applied to ss_Delta. It is entirely optional to apply this to a custom table or against the full SF object name

```
exec ss_Delta 'DEMO', 'Opportunity', null, 'Table:OppTest','Where StageName = ''Closed Won'''
```

Note, any custom Replica created with a Where clause must continue to have that same identical Where clause applied any subsequent ss_Delta calls, otherwise the Delta logic will be compromised as there will be no consistency between the scope of the original ss_Replica and the delta data retrievals via ss_Delta.

In this example, a where clause has been added to the base Salesforce object replica (i.e. no custom table usage), the clause Type = New Customer has been passed in

```
exec ss_Replica 'DEMO', 'Opportunity',null,'Where Type = ''New Customer'''
exec ss_Delta 'DEMO', 'Opportunity',null,null,'Where Type = ''New Customer'''
```

Combining Top with Custom Replica and where clause

```
exec ss_Replica 'DEMO', 'Opportunity:Top20','Table:OppTest','Where StageName = ''Closed Won'''
```

Combining Top with Field Subset, Custom Replica and where clause

```
exec ss_Replica 'DEMO', 'Opportunity:Top20','Subset(Name,Type,StageName) Table:OppTest','Where
StageName = ''Closed Won'''
```

ContentDocumentLink

The Salesforce Content Model object acts as a junction between a Document and the object (for example Opportunity) that has the Document linked to it. It allows multiple different objects to be linked to the same Document.

Field	Purpose
LinkedEntityId	Object Id (for example Account.Id; Opportunity.Id) etc
ContentDocumentId	Document Id

The Salesforce api limits the retrieving of ContentDocumentLink records to either a maximum of 200 LinkedEntityId or 200 ContentDocumentId. For this reason replicating ContentDocumentLink can be challenging, however SQL-Sales is able to fully return all available records en masse using the conventional ss_Replica command:

```
exec ss_Replica 'DEMO', 'ContentDocumentLink'
```

if specific subsets of either LinkedEntityId or ContentDocumentId are required (in batches of 200),

ss_Replica allows you to pass in a comma separated string of up to 200 Salesforce 18 character Ids into optional parameter @Special2. @Special1 is how you indicate if you are passing in LinkedEntityId Ids or ContentDocumentId Ids. Note this is supported purely as a helper shortcut to retrieve via the Salesforce api limits on querying ContentDocumentLink.

Example of usage (LinkedEntityId)

@Special1 in this example is 'LinkedEntityId'

@Special2 in this example is the variable [@Ids] which has the comma separated Ids for example up to 200 Opportunity Ids

```
exec ss_Replica 'DEMO', 'ContentDocumentLink','LinkedEntityId',@Ids
```

As a helper, below are three suggested methods by which you can create a comma separated string for which can be passed into the variable @Ids

Note, in the below examples of 200 Account Ids, Account has been prior replicated

Option1 (COALESCE)

```
DECLARE @Ids NVARCHAR(4000)
SELECT @Ids = COALESCE(@Ids + ',', '') + CAST(Id AS VARCHAR)
FROM (SELECT TOP 200 Id FROM Account) AS Top200
exec ss_Replica 'DEMO', 'ContentDocumentLink','LinkedEntityId',@Ids
```

Option2 (STUFF | XML PATH)

```
DECLARE @Ids NVARCHAR(4000)
SELECT @Ids = (       STUFF((SELECT TOP 200 ',' + CAST(Id AS VARCHAR)
FROM Account ORDER BY Id FOR XML PATH('')), 1, 1, ''))
exec ss_Replica 'DEMO', 'ContentDocumentLink','LinkedEntityId',@Ids
```

Option3 (STRING_AGG)

```
DECLARE @Ids NVARCHAR(4000)
SELECT @Ids = (       SELECT STRING_AGG(CAST(Id AS VARCHAR), ',') WITHIN GROUP (ORDER BY Id) AS
CommaSeparatedIds
FROM (SELECT TOP 200 Id FROM Account ORDER BY Id) AS Top200)
exec ss_Replica 'DEMO', 'ContentDocumentLink','LinkedEntityId',@Ids
```

Example of usage (ContentDocumentId)

@Special1 in this example is 'ContentDocumentId'

@Special2 in this example is the variable [@Ids] which has the comma separated Ids for example up to 200 Opportunity Ids

```
exec ss_Replica 'DEMO', 'ContentDocumentLink','ContentDocumentId',@Ids
```

As a helper, below are three suggested methods by which you can create a comma separated string for which can be passed into the variable @Ids

Note, in the below examples of 200 Document Ids, ContentVersion has been prior replicated

Option1 (COALESCE)

```
DECLARE @Ids NVARCHAR(4000)
SELECT @Ids = COALESCE(@Ids + ',', '') + CAST(ContentDocumentId AS NCHAR(18))
FROM (SELECT DISTINCT TOP 200 ContentDocumentId FROM ContentVersion) AS Top200
exec ss_Replica 'DEMO', 'ContentDocumentLink','ContentDocumentId',@Ids
```

Option2 (STUFF | XML PATH)

```
DECLARE @Ids NVARCHAR(4000)
SELECT @Ids = (       STUFF((SELECT TOP 200 ',' + CAST(ContentDocumentId AS NCHAR(18))
FROM ContentVersion ORDER BY ContentDocumentId FOR XML PATH('')), 1, 1, ''))
exec ss_Replica 'DEMO', 'ContentDocumentLink','ContentDocumentId',@Ids
```

Option3 (STRING_AGG)

```
DECLARE @Ids NVARCHAR(4000)
SELECT @Ids = (       SELECT STRING_AGG(CAST(ContentDocumentId AS NCHAR(18)), ',') WITHIN GROUP
(ORDER BY ContentDocumentId) AS CommaSeparatedIds
FROM (SELECT TOP 200 ContentDocumentId FROM ContentVersion ORDER BY ContentDocumentId) AS Top200)
exec ss_Replica 'DEMO', 'ContentDocumentLink','ContentDocumentId',@Ids
```

ss_ReplicaAll

ss_ReplicaAll allows you to run ss_Replica in bulk.

Simple Example

```
exec ss_ReplicaAll 'DEMO'
```

With @ObjectNameFrom

```
exec ss_ReplicaAll 'DEMO', 'Opportunity'
```

This will commence the bulk ss_Replica replication from the Object immediately following Opportunity, which could be OpportunityCompetitor if there is no other custom or standard object with a name closer to Opportunity, sorting alphabetically. This is useful if a prior run of ss_ReplicaAll has failed for whatever reason and you wish to re-commence from the point of failure.

Parameters

Parameter	Purpose
@Env	SQL Sales Environment Name
@ObjectNameFrom	This will commence the bulk ss_Replica replication from the Object immediately following the one passed in via @ObjectNameFrom. This is useful if a prior run of ss_ReplicaAll has failed for whatever reason and you wish to re-commence from the point of failure.

ss_ObjectInclusion inclusion table

Field	Purpose
ObjectName	Definition of Objects you wish to include in each ss_ReplicaAll run. Only Objects contained in this table will be considered in scope. Accompanying table ss_ObjectExclusion will additionally be referenced, i.e. Objects within will be ignored.

ss_ObjectInclusion exclusion table

Field	Purpose
ObjectName	Add any Objects in this table to have them excluded from each ss_ReplicaAll run. On deployment (ss_EnableDatabase) a number of known problem Objects are included in this table along with their API Response Error Msg and Codes, for general reference. It is a matter of choice whether you specify an Inclusion set (via ss_ObjectInclusion) or exclude via this table. Generally, only a relatively small number of Salesforce objects are required to maintain a working replication database and hence ss_ObjectInclusion is more appropriate, or even a hard coded list of ss_Delta commands for each required Object, for example: ss_Replica 'DEMO', 'Account' ss_replica 'DEMO', 'AccountHistory' ss_replica 'DEMO', 'Opportunity' Which is run via a SQL Agent Job or similar. Running with no entries in this table but some entries in ss_ObjectExclusion would be more appropriate, if you wanted to automatically take an entire backup of your Salesforce Org, for example.
ErrorMsg	Salesforce API Response Error Msg
ErrorCode	Salesforce API Response Error code

DELTA REPLICATION

ss_Delta

ss_Delta is a Stored Procedure you execute within your enabled database.

Prerequisites

- A working Environment and the SQL Sales Daemon is running (see Environment Setup)
- SQL Server
- Compiled stored procedure ss_Delta (see Database Enabling)

ss_Log replication log

Replication is logged in table "ss_Log" which is automatically initially created by a prior ss_Replica runs and maintained for all subsequent ss_Delta runs.

Note, the table is created within the schema to which you are running ss_Delta, hence if you ran with uat.ss_Delta, the resultant entry for the given object will be to table uat.ss_Log.

Field	Purpose
ReplicaName	The Salesforce object name or a Custom table name if one has been specified, created as a replica table in the given enabled Database
ObjectName	Source Salesforce Object of this replication log (will be the same as ReplicaName if no customisation has occurred with the "Table:" switch
CustomReplica	Indicates if the Replica is Custom (i.e. not a basic replication of a Salesforce Object Name but ReplicaName is a custom table name
TableCreatedDate	Serves no functional purpose, is for information purposes only
MaxSystemDate	This is how the ss_Delta process determines how to delta replicate
LogDate	Last log datetime
TopRow	Captures any Top commands, for example 'Account:Top100'
Subset	Captures any Column Subset commands, for example 'Subset(Name,BillingCountry)'
Status	Possible values: FAILURE (indicates a failure has occurred, see Detail field) FULL (indicates a Full replication occurred, via ss_Replica) DELTA (indicates a partial replication occurred)
WhereInput	Captures any Where clause commands injected into the replication
Detail	Success or Failure detail

ss_Log_Working detailed replication log

For more granular detail of every replication run, each step is logged in "ss_Log_Working" which can be helpful in monitoring longer running processes of millions of rows although for ss_Delta runs that is a less common scenario, unless significant changes have occured on the given Salesforce Object – if so it may be more efficient in that situation to run ss_Replica. The BULK connection method will deliver the data payload in chunks, which are logged and so can be monitored to check progress.

Field	Purpose
Environment	The specified Salesforce Environment, defined in the SQL Sales Configuration tool
Task	"ss_Delta"
ObjectName	Source Salesforce Object of this replication log (will be the same as ReplicaName if no customisation has occurred with the "Table:" switch
PreProcessPoint	The process which is about to be started
Detail1	The contents of parameter @Special1
Detail2	The contents of parameter @Special2
Detail3	The contents of parameter @AutoReplica
Detail4	Logs the current Chunk, if a Salesforce Id, this is the latest Id the process has got to

Note:

1. The table is created within the schema to which you are running ss_Delta, hence if you ran with uat.ss_Delta, the resultant entry for the given object will be to table uat.ss_Log / uat.ss_Log_Working.
2. ss_Delta will not run if no prior entry exists in the ss_Log table (unless 'Yes' was passed to parameter @AutoReplica).
3. If an entry does exist in the ss_Log table but the last run was greater than 7 days ago, ss_Delta will not run, however if input parameter @AutoReplica = Yes then in this event, the ss_Delta run will failover to running ss_Replica, passing in the same input parameters as provided in the ss_Delta execution.

Parameters

Parameter	Purpose
@Env	SQL Sales Environment Name
@ObjectName	Salesforce Object Name (for example Account) Can be accompanied with :**Top**XXX where XXX is a number exec ss_Delta 'DEMO', 'Account:Top10' For example Account:Top10
@AutoReplica	**Full** = if the given Salesforce object's meta data definition is identified to be different to the local definition (in the prior replicated SQL table), if "Full" is set in this parameter then ss_Replica will automatically run to fully replicate. If no value is passed, (by default the setting is "No"), this instructs ss_Delta to ignore any potentially new fields or different field definitions for existing fields. Any Failure scenarios will also failover to a full replication via ss_Replica if this parameter is set to "Full", for example: Objects that cannot technically be delta replicated If the given Object has not yet been fully replicated via ss_Replica If the entry in ss_Log is not present (yet the SQL table does exist) If the last replication was > 7 days ago
@Special1	Optional, in simple usage, null Options **Subset(field1,field2,field3)** – where field1 etc are Salesforce api fieldnames for the given object. For example in the case of the object being Account, this could be: Subset(Name,Phone,BillingCity) **Table:<custom table name>** - for example 'Table:OppTest' No special delimiter is required between Subset(x,y,z) and Table:xxx, for example leave nothing or a space: 'Subset(Name,Phone,BillingCity) Table:AccountTest'
@Special2	Optional, in simple usage, null **Where <SOQL>** - for example 'Where StageName = ''Closed Won'''

AutoReplica

Full = if the given Salesforce object's meta data definition is identified to be different to the local definition (in the prior replicated SQL table), if "Full" is set in this parameter then ss_Replica will automatically run to fully replicate. If no value is passed, (by default the setting is "No"), this instructs ss_Delta to ignore any potentially new fields or different field definitions for existing fields.

Any Failure scenarios will also failover to a full replication via ss_Replica if this parameter is set to "Full", for example:

- Objects that cannot technically be delta replicated
- If the given Object has not yet been fully replicated via ss_Replica
- If the entry in ss_Log is not present (yet the SQL table does exist)
- If the last replication was > 7 days ago

Input parameters provided to ss_Delta will be passed through to ss_Replica on an AutoReplica failover

For example taking this fairly involved example of a ss_Replica:

```
exec ss_Replica 'DEMO', 'Account', 'Subset(Name) Table:CustomAcc','Where Name like ''%plc%'''
```

The subsequent ss_Delta of:

```
exec ss_Delta 'DEMO', 'Account', 'Full', 'Subset(Name) Table:CustomAcc','Where Name like
''%plc%'''
```

In the event of any of the conditions for a failover being met, will have @Special1 and @Special2 passed through to the called ss_Replica, i.e. for this example:

@Special1 = 'Subset(Name) Table:CustomAcc'

@Special2 = 'Where Name like ''%plc%'''

Table ss_AutoReplica

On deployment table ss_AutoReplica is populated with Object Names of Objects that must be fully replicated (i.e. where it is not appropriate to Delta Replicate). If you run ss_Delta with @AutoReplica = Full then any Object in this table will automatically be fully replicated, hence you can leverage this table to suit your own requirements if you require a given Object to always fully replicate, provided it is present in table ss_AutoReplica and parameter @AutoReplica = Full.

Field	Purpose
ObjectName	For the given Object, will force a full replication, via ss_Replica, if parameter @AutoReplica = Full.

Simple Example

```
exec ss_Delta 'DEMO', 'Account'
```

Specifying Batchsize

By default ss_Delta will use a batchsize of 2000 (the maximum possible via the Salesforce SOAP api). Should you wish to use a smaller batchsize, specify this by passing in the custom batchsize when passing in the Objectname as below:

```
exec ss_Delta 'DEMO', 'Account(25)'
```

Subset Example

```
exec ss_Delta 'DEMO', 'Account', null, 'Subset(Name,BillingCountry)'
```

Valid fields (i.e. that exist on the specified object and for which you have permission to query, will only be returned, although note some basic fields to support the replication logic to function correctly will be imposed (for example Id, isDeleted; SystemModStamp).

Use subset with caution as it will only delta maintain the subset of columns specified, which when run against a fully replication table, can leave fields not in the subset scope, stale. Generally you would use the same subset scope of fields in the initial ss_Replica and all subsequent ss_Delta runs.

Note, for larger sets of subset fields, you may prefer to input in a list form, example shown below, which is an acceptable input method to the @Special1 input parameter:

```
exec ss_Delta 'DEMO', 'Opportunity', null
,'Subset(
AccountId
,Amount
,CampaignId
,CloseDate
,Name
,NextStep
,OrderNumber__c
,OwnerId
,Pricebook2Id
,Probability
,StageName
,SystemModstamp
,TotalOpportunityQuantity
,TrackingNumber__c
,Type)'
```

Or

```
exec ss_Delta 'DEMO', 'Opportunity', null
,'Subset(
AccountId,
Amount,
CampaignId,
CloseDate,
Name,
NextStep,
OrderNumber__c,
OwnerId,
Pricebook2Id,
Probability,
StageName,
SystemModstamp,
TotalOpportunityQuantity,
TrackingNumber__c,
Type)'
```

Schema | Metadata change

Any changes to Salesforce since the last ss_Replica full rebuild (i.e. the local replica table), will be notified in the output report when running ss_Delta. In the example below the field "TestCheckbox__c" has been identified as being newly available. ss_Delta makes no attempt to merge this into the local, as there will inevitably be rows in the local replica that fall outside the delta scope of the current delta replication. It is therefore recommended to run a full ss_Replica at the next available opportunity. Unless you have run ss_Delta with @AutoReplica = 'Full' in which case ss_Replica will have been automatically run and you wouldn't be encountering this information message

```
21:43:49: New Column: TestCheckbox__c has subsequently been created/made visible in SF since the
last full replication, therefore excluded from delta scope: ss_replica recommended!
```

Combined Subset, Batchsize and Top check example

```
exec ss_Delta 'DEMO', 'Account(25):Top100', null, 'Subset(Name,BillingCountry)'
```

Custom Replica Example

```
exec ss_Delta 'DEMO', 'Opportunity', null, 'Table:OppTest'
```

In this example, Salesforce Opportunity data will be delta replicated to the prior created via ss_Rplica "OppTest" SQL table. This will exist entirely independently of any potential "Opportunity" replica existing in the same deployed schema in your given Database.

For example running

```
exec ss_Delta 'DEMO', 'Opportunity', null, 'Table:OppTest'
exec ss_Delta 'DEMO', 'Opportunity'
```

will maintain two separate replica tables of Salesforce Opportunity data, one called Opportunity, the other called OppTest. Note, the "Table:" switch must be applied consistently on each subsequent ss_Delta run.

Where Clause Example

```
exec ss_Delta 'DEMO', 'Opportunity', null, 'Table:OppTest','Where StageName = ''Closed Won'''
```

In this example, a where clause has been added, which will be applied to the delta replica. It is entirely optional to apply this to a custom table or against the full SF object name

Note, any custom Replica created with a Where clause must continue to have that same identical Where clause applied any subsequent ss_Delta calls, otherwise the Delta logic will be compromised as there will be no consistency between the scope of the original ss_Replica and the delta data retrievals via ss_Delta.

In this example, a where clause has been added to the base Salesforce object replica (i.e. no custom table usage), the clause Type = New Customer has been passed in

```
exec ss_Replica 'DEMO', 'Opportunity',null,'Where Type = ''New Customer'''
exec ss_Delta 'DEMO', 'Opportunity',null,null,'Where Type = ''New Customer'''
```

ss_DeltaAll

ss_DeltaAll allows you to run ss_Delta in bulk.

Simple Example

```
exec ss_DeltaAll 'DEMO'
```

With @ObjectNameFrom

```
exec ss_DeltaAll 'DEMO', 'Opportunity'
```

This will commence the bulk ss_Delta replication from the Object immediately following Opportunity, which could be OpportunityCompetitor if there is no other custom or standard object with a name closer to Opportunity, sorting alphabetically. This is useful if a prior run of ss_DeltaAll has failed for whatever reason and you wish to re-commence from the point of failure.

With AutoReplica = Full

```
exec ss_DeltaAll 'DEMO',null,'Full'
```

@AutoReplica will be passed through to each attempted ss_Delta, which will automatically failover to ss_Replica should any AutoReplica condition be met.

Parameters

Parameter	Purpose
@Env	SQL Sales Environment Name
@ObjectNameFrom	This will commence the bulk ss_Delta replication from the Object immediately following the one passed in via @ObjectNameFrom. This is useful if a prior run of ss_DeltaAll has failed for whatever reason and you wish to re-commence from the point of failure.
@AutoReplica	@AutoReplica will be passed through to each attempted ss_Delta, which will automatically failover to ss_Replica should any AutoReplica condition be met.

ss_ObjectInclusion inclusion table

Field	Purpose
ObjectName	Definition of Objects you wish to include in each ss_DeltaAll run. Only Objects contained in this table will be considered in scope. Accompanying table ss_ObjectExclusion will additionally be referenced, i.e. Objects within will be ignored.

ss_ObjectInclusion exclusion table

Field	Purpose
ObjectName	Add any Objects in this table to have them excluded from each ss_DeltaAll run. On deployment (ss_EnableDatabase) a number of known problem Objects are included in this table along with their API Response Error Msg and Codes, for general reference. It is a matter of choice whether you specify an Inclusion set (via ss_ObjectInclusion) or exclude via this table. Generally, only a relatively small number of Salesforce objects are required to maintain a working replication database and hence ss_ObjectInclusion is more appropriate, or even a hard coded list of ss_Delta commands for each required Object, for example: ss_Delta 'DEMO', 'Account' ss_Delta 'DEMO', 'AccountHistory' ss_Delta 'DEMO', 'Opportunity' Which is run via a SQL Agent Job or similar. Running with no entries in this table but some entries in ss_ObjectExclusion would be more appropriate, if you wanted to automatically take an entire backup of your Salesforce Org, for example.
ErrorMsg	Salesforce API Response Error Msg
ErrorCode	Salesforce API Response Error code

META DATA

ss_MetaObject

Object Metadata

Pull all the objects in your Salesforce instance, or optionally just a single specified one.

Results are loaded to table `ss_SysObject`

Parameters

Parameter	Purpose
@ObjectName	Salesforce Object Name (for example Account). alternatively pass "All" to retrieve all Objects.
@Env	SQL Sales Environment Name, see Setup
@Special1	Future use, not currently used
@Special2	Future use, not currently used

Example of Use – All

```
exec ss_MetaObject 'DEMO', 'All'
```

```
SQL-SALES ss_SysObject run date: 2023-11-09 ----------------------
23:04:16: Using Env|Schema: DEMO|dbo
23:04:16: Starting ss_SysObject retrieval for: All Objects
23:04:19: Connection method BULK & SOAP API
23:04:19: Drop existing ss_SysObject if exists and recreate
23:04:21: Load to dbo.ss_SysObject
------------------------------------------------------------------
```

```
select * from ss_SysObject
```

Messages

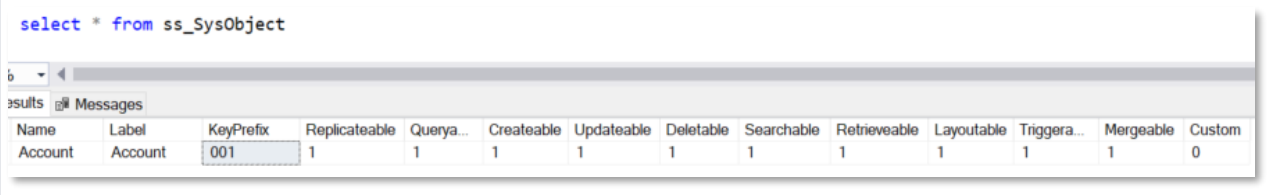
Name	Label	KeyPrefix	Replicateable	Querya...	Createable	Updateable	Deletable	Searchable	Retrieveable	Layoutable	Triggera...	Mergeable	Custom
AIApplication	AI Application	0Pp	0	1	0	0	0	0	1	0	0	0	0
AIApplicationConfig	AI Application config	6S9	0	1	0	0	0	0	1	0	0	0	0
AIInsightAction	AI Insight Action	9qd	1	1	0	0	0	0	1	0	0	0	0
AIInsightFeedback	AI Insight Feedback	9bq	1	1	0	0	0	0	1	0	0	0	0
AIInsightReason	AI Insight Reason	0T2	1	1	0	0	0	0	1	0	0	0	0
AIInsightValue	AI Insight Value	9qc	1	1	0	0	0	0	1	0	0	0	0
AIPredictionEvent	AI Prediction Event	0ae	0	0	0	0	0	0	0	0	1	0	0
AIRecordInsight	AI Record Insight	9qb	1	1	0	0	1	0	1	0	1	0	0
AcceptedEventRelation	Accepted Event Relation	NULL	0	1	0	0	0	0	1	0	0	0	0
Account	Account	001	1	1	1	1	1	1	1	1	1	1	0
AccountChangeEvent	Account Change Event	NULL	0	0	0	0	0	0	1	0	1	0	0

Example of Use – Single

```
exec ss_MetaObject 'DEMO', 'Account'
```

```
SQL-SALES ss_SysObject run date: 2023-11-09 ----------------------
```

```
23:06:59: Using Env|Schema: DEMO|dbo
23:06:59: Starting ss_SysObject retrieval for: Account
23:07:02: Connection method BULK & SOAP API
23:07:02: Drop existing ss_SysObject if exists and recreate
23:07:04: Load to dbo.ss_SysObject
------------------------------------------------------------------
```



```
select * from ss_SysObject
```

Messages

Name	Label	KeyPrefix	Replicateable	Querya...	Createable	Updateable	Deletable	Searchable	Retrieveable	Layoutable	Triggera...	Mergeable	Custom
Account	Account	001	1	1	1	1	1	1	1	1	1	1	0

ss_SysObject table

- Name (api Name)
- Label (Label)
- KeyPrefix (where available/relevant, Object Id first 3 characters)
- Replicateable (boolean)
- Queryable (boolean)
- Createable (boolean)
- Updateable (boolean)
- Deletable (boolean)
- Searchable (boolean)
- Retrieveable (boolean)
- Layoutable (boolean)
- Triggerable (boolean)
- Mergeable (boolean)
- Custom (boolean)

ss_MetaField

Field Metadata

Pull all the fields in your Salesforce instance, per Object, or optionally just a single specified Object's fields.

Results are loaded to table `ss_SysField`

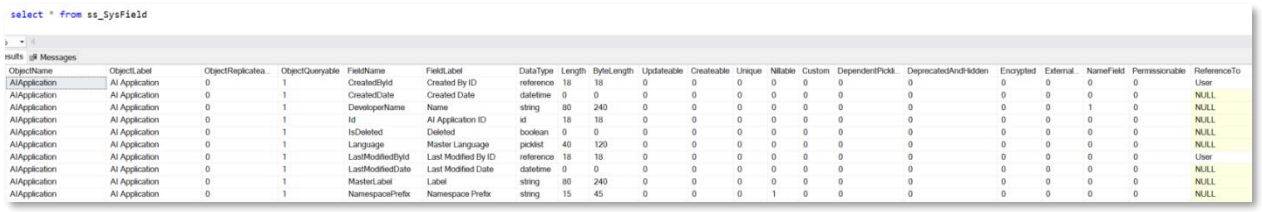
Parameters

Parameter	Purpose
@ObjectName	Salesforce Object Name (for example Account). alternatively pass "All" to retrieve all Objects.
@Env	SQL Sales Environment Name, see Setup
@Special1	Optional, default is "B" for BULK. This allows the Environment "Connection Method" setting to be overridden, per ss_MetaField run. Pass "O" for ODBC or "B" for BULK.
@Special2	Future use, not currently used

Example of Use - All

```
exec ss_MetaField 'DEMO', 'All'
```

```
SQL-SALES ss_SysField run date: 2023-11-09 ----------------------
23:15:07: Using Env|Schema: DEMO|dbo
23:15:07: Starting ss_SysField retrieval for: All Objects
23:15:09: Connection method BULK & SOAP API
23:15:09: Drop existing ss_SysField if exists and recreate
23:16:57: Load to dbo.ss_SysField
------------------------------------------------------------------
```



```
select * from ss_SysField
```

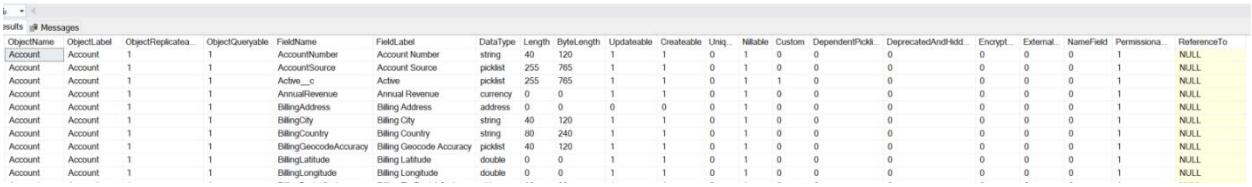
ObjectName	ObjectLabel	ObjectReplicatea...	ObjectQueryable	FieldName	FieldLabel	DataType	Length	ByteLength	Updateable	Createable	Unique	Nillable	Custom	DependentPicklI...	DeprecatedAndHidden	Encrypted	External...	NameField	Permissionable	ReferenceTo
AIApplication	AI Application	0	1	CreatedById	Created By ID	reference	18	18	0	0	0	0	0	0	0	0	0	0	0	User
AIApplication	AI Application	0	1	CreatedDate	Created Date	datetime	0	0	0	0	0	0	0	0	0	0	0	0	0	NULL
AIApplication	AI Application	0	1	DeveloperName	Name	string	80	240	0	0	0	0	0	0	0	0	0	1	0	NULL
AIApplication	AI Application	0	1	Id	AI Application ID	id	18	18	0	0	0	0	0	0	0	0	0	0	0	NULL
AIApplication	AI Application	0	1	IsDeleted	Deleted	boolean	0	0	0	0	0	0	0	0	0	0	0	0	0	NULL
AIApplication	AI Application	0	1	Language	Master Language	picklist	40	120	0	0	0	0	0	0	0	0	0	0	0	NULL
AIApplication	AI Application	0	1	LastModifiedById	Last Modified By ID	reference	18	18	0	0	0	0	0	0	0	0	0	0	0	User
AIApplication	AI Application	0	1	LastModifiedDate	Last Modified Date	datetime	0	0	0	0	0	0	0	0	0	0	0	0	0	NULL
AIApplication	AI Application	0	1	MasterLabel	Label	string	80	240	0	0	0	0	0	0	0	0	0	0	0	NULL
AIApplication	AI Application	0	1	NamespacePrefix	Namespace Prefix	string	15	45	0	0	0	1	0	0	0	0	0	0	0	NULL

Example of Use - Single

```
exec ss_MetaField 'DEMO', 'Account'
```

```
SQL-SALES ss_SysField run date: 2023-11-09 ----------------------
23:22:56: Using Env|Schema: DEMO|dbo
23:22:56: Starting ss_SysField retrieval for: Account
23:22:58: Connection method BULK & SOAP API
23:22:58: Drop existing ss_SysField if exists and recreate
23:23:00: Load to dbo.ss_SysField
------------------------------------------------------------------
```

```
select * from ss_SysField
```



ObjectName	ObjectLabel	ObjectReplicatea...	ObjectQueryable	FieldName	FieldLabel	DataType	Length	ByteLength	Updateable	Createable	Uniq...	Nillable	Custom	DependentPickli...	DeprecatedAndHidd...	Encrypt...	External...	NameField	Permissiona...	ReferenceTo
Account	Account	1	1	AccountNumber	Account Number	string	40	120	1	1	0	1	0	0	0	0	0	0	1	NULL
Account	Account	1	1	AccountSource	Account Source	picklist	255	765	1	1	0	1	0	0	0	0	0	0	1	NULL
Account	Account	1	1	Active__c	Active	picklist	255	765	1	1	0	1	1	0	0	0	0	0	1	NULL
Account	Account	1	1	AnnualRevenue	Annual Revenue	currency	0	0	1	1	0	1	0	0	0	0	0	0	1	NULL
Account	Account	1	1	BillingAddress	Billing Address	address	0	0	0	0	0	1	0	0	0	0	0	0	1	NULL
Account	Account	1	1	BillingCity	Billing City	string	40	120	1	1	0	1	0	0	0	0	0	0	1	NULL
Account	Account	1	1	BillingCountry	Billing Country	string	80	240	1	1	0	1	0	0	0	0	0	0	1	NULL
Account	Account	1	1	BillingGeocodeAccuracy	Billing Geocode Accuracy	picklist	40	120	1	1	0	1	0	0	0	0	0	0	1	NULL
Account	Account	1	1	BillingLatitude	Billing Latitude	double	0	0	1	1	0	1	0	0	0	0	0	0	1	NULL
Account	Account	1	1	BillingLongitude	Billing Longitude	double	0	0	1	1	0	1	0	0	0	0	0	0	1	NULL

ss_SysField table

- ObjectName (Object api Name)
- ObjectLabel (Object Label)
- ObjectReplicateable (boolean)
- ObjectQueryable (boolean)
- FieldName (Field api Name)
- FieldLabel (Field Label)
- DataType (Field data type)
- Length (Field data length)
- ByteLength (Field byte size)
- Updateable (boolean)
- Createable (boolean)
- Unique (boolean)
- Nillable (boolean)
- Custom (boolean)
- DependentPicklist (boolean)
- DeprecatedAndHidden (boolean)
- Encrypted (boolean)
- ExternalId (boolean)
- NameField (boolean)
- Permissionable (boolean)
- ReferenceTo (Related Object(s))

ss_MetaPick

Picklist Metadata

Pull all Picklist values, per Object, or optionally just a single specified Object's picklists.

Results are loaded to table `ss_SysPicklist`

Parameters

Parameter	Purpose
@ObjectName	Salesforce Object Name (for example Account). alternatively pass "All" to retrieve all Objects.
@Env	SQL Sales Environment Name, see Setup
@Special1	Future use, not currently used
@Special2	Future use, not currently used

Example of Use - All

```
exec ss_MetaPick 'DEMO', 'All'
```

```
SQL-SALES ss_SysPicklist run date: 2023-11-09 --------------------
23:29:32: Using Env|Schema: DEMO|dbo
23:29:32: Starting ss_SysPicklist retrieval for: All Objects
23:29:34: Connection method BULK & SOAP API
23:29:34: Drop existing ss_SysPicklist if exists and recreate
23:30:57: Load to dbo.ss_SysPicklist
------------------------------------------------------------------
```

```
select * from ss_SysPicklist
```

esults Messages

ObjectName	FieldName	Value	Label	Active	DefaultValue
AIRecordInsight	Type	Lookup	Lookup	1	0
AIRecordInsight	Type	MultiValue	MultiValue	1	0
AIRecordInsight	Type	SimilarRecord	SimilarRecord	1	0
AIRecordInsight	Type	SingleValue	SingleValue	1	0
Account	AccountSource	Other	Other	1	0
Account	AccountSource	Partner Referral	Partner Referral	1	0
Account	AccountSource	Phone Inquiry	Phone Inquiry	1	0
Account	AccountSource	Purchased List	Purchased List	1	0
Account	AccountSource	Web	Web	1	0
Account	Active__c	No	No	1	0

Example of Use - Single

```
exec ss_MetaPick 'DEMO', 'Account'
```

```
SQL-SALES ss_SysPicklist run date: 2023-11-09 -------------------
23:37:52: Using Env|Schema: DEMO|dbo
23:37:52: Starting ss_SysPicklist retrieval for: Account
23:37:55: Connection method BULK & SOAP API
23:37:55: Drop existing ss_SysPicklist if exists and recreate
23:37:58: Load to dbo.ss_SysPicklist
-----------------------------------------------------------------
```

ss_SysPicklist table

- ObjectName (Object api Name)
- FieldName (Field api Name)
- Value (Picklist Value)
- Label (Picklist Label)
- Active (boolean)
- DefaultValue (boolean)

HELPER TOOLS

ss_UserInfo

User Information

Results are loaded to table `ss_SysUserInfo`

> Note, if OAuth2.0 has been configured as your Connection Method in the Environment Configuration tool, the User Info returned will be for the defined “Integration Username”

Parameters

Parameter	Purpose
@Env	SQL Sales Environment Name, see Setup
@Special1	Future use, not currently used
@Special2	Future use, not currently used

Example of Use

```
exec ss_UserInfo 'DEMO'
```

```
SQL-SALES ss_SysUserInfo run date: 2023-11-09 ----------------------
23:51:09: Using Env|Schema: DEMO2|dbo
23:51:09: Starting ss_SysUserInfo retrieval
23:51:12: Connection method ODBC & SOAP API
23:51:12: Drop existing ss_SysUserInfo if exists and recreate
23:51:14: Load to dbo.ss_SysUserInfo
------------------------------------------------------------------
```

ss_SysUserInfo table

- UserId (User.Id of the @Env user)
- Name (User.Name of the @Env user)
- Username (User.Username of the @Env user)
- Email (User.Email of the @Env user)
- ProfileName (User.ProfileId.Profile.Name of the @Env user)
- RoleName (User.RoleId.UserRole.Name of the @Env user)

- SessionId (The Salesforce Session Id of the established session)
- UserGroups (Member Groups of the @Env user)
- PermissionSetGroups (PermissionSetGroups of the @Env user)

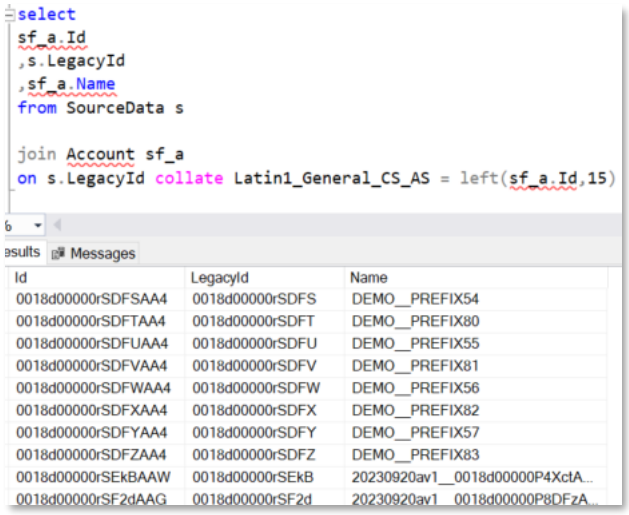
Working with Salesforce 15 character Ids

If you have a requirement to bulk query source 15 character Ids against SQL Sales replicated 18 character Ids, you are recommended to force a case sensitive (CS) collation in the query.

For example:

```
select
sf_a.Id
,s.LegacyId
,sf_a.Name
from SourceData s

join Account sf_a
on s.LegacyId collate Latin1_General_CS_AS = left(sf_a.Id,15)
```

Here, the Sourcedata.LegacyId is a case sensitive 15 character Salesforce Id. Assuming your database collation is case insensitive (which is generally the case), it is unsafe to assume a simple left(Id,15) will reliably return a single match, as for example 0018d00000rRRGeAAO and 0018d00000rRRgeAAO are different Ids, yet left(Id,15) in a default case insensitive collation will regard them as the same string.



```
select
sf_a.Id
,s.LegacyId
,sf_a.Name
from SourceData s

join Account sf_a
on s.LegacyId collate Latin1_General_CS_AS = left(sf_a.Id,15)
```

esults | Messages

Id	LegacyId	Name
0018d00000rSDFSAA4	0018d00000rSDFS	DEMO__PREFIX54
0018d00000rSDFTAA4	0018d00000rSDFT	DEMO__PREFIX80
0018d00000rSDFUAA4	0018d00000rSDFU	DEMO__PREFIX55
0018d00000rSDFVAA4	0018d00000rSDFV	DEMO__PREFIX81
0018d00000rSDFWAA4	0018d00000rSDFW	DEMO__PREFIX56
0018d00000rSDFXAA4	0018d00000rSDFX	DEMO__PREFIX82
0018d00000rSDFYAA4	0018d00000rSDFY	DEMO__PREFIX57
0018d00000rSDFZAA4	0018d00000rSDFZ	DEMO__PREFIX83
0018d00000rSEkBAAW	0018d00000rSEkB	20230920av1__0018d00000P4XctA...
0018d00000rSF2dAAG	0018d00000rSF2d	20230920av1__0018d00000P8DFzA...

If you have a need to spot check the conversion from a 15 character to 18 character, refer to the ss_18 function which follows, although for performance reasons generally speaking the collation technique described here is recommended.

ss_18

15 to 18 Salesforce Id conversion

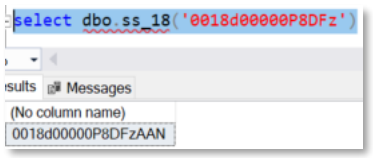
Simple function to return the case insensitive 18 character Salesforce Id from an inputted 15 character case sensitive Id.

Parameters

Parameter	Purpose
@ui_id	nchar(15) inputted Salesforce 15 character Id

Example of Use

```
select dbo.ss_18('0018d00000P8DFz')
```



ss_Admin

Stopping the Daemon

```
exec ss_Admin 'STOP'
```

```
SQL-SALES ss_Admin run date: 2023-11-09 ----------------------
23:42:30: Using Schema: dbo
23:42:31: SQL-Sales Daemon successfully stopped
-------------------------------------------------------------
```

Starting the Daemon

```
exec ss_Admin 'START'
```

```
SQL-SALES ss_Admin run date: 2023-11-09 ----------------------
23:43:20: Using Schema: dbo
23:43:27: SQL-Sales Daemon successfully started
-------------------------------------------------------------
```

Stopping the Handler

```
exec ss_Admin 'STOP_HANDLER'
```

```
SQL-SALES ss_Admin run date: 2023-11-09 ----------------------
23:44:40: Using Schema: dbo
23:44:41: SQL-Sales Handler successfully stopped
-------------------------------------------------------------
```

Checking the Environments

```
exec ss_Admin 'ENVS'
```

```
SQL-SALES ss_Admin run date: 2023-11-09 ----------------------
23:47:07: Using Schema: dbo
23:47:08: Configured Environments, Name (Username):
23:47:08: DEMO2 (demo2@sql-sales.com)
23:47:08: DEMO (demo@sql-sales.com)
23:47:08: HARD (hard.delete@sql-sales.com)
-------------------------------------------------------------
```

LOADING (SS_LOADER)

SQL Sales has a powerful yet simple to use feature supporting the following data operations, designed to make working with Salesforce very straightforward for those familiar and comfortable using SQL Server. ss_loader works with the standard web services SOAP API, including loading Files & Notes. It also supports both Version 1 and Version 2 of the Bulk API. These Salesforce APIs do not support all data operations, the supported set, per API are as below:

SOAP API	BULK API v1	BULK API v2
Insert	BulkAPIv1Insert	BulkAPIv2Insert
Update	BulkAPIv1Update	BulkAPIv2Update
Delete	BulkAPIv1Delete	BulkAPIv2Delete
Undelete	n/a	n/a
Upsert	BulkAPIv1Upsert	BulkAPIv2Upsert
n/a	BulkAPIv1Harddelete	BulkAPIv2Harddelete

Provide your data in a SQL table and point to Salesforce via a configured Environment, choosing one of the supported operations. Success is logged in the required Error field, or any failures passed back via the Salesforce api are also provided in the Error field.

In the case of Inserts, the newly created Id is helpfully populated into the Id column. Similarly in the case of Upsert-Inserts – i.e. if no match is found via the specified External Id and an Insert occurs, the new Id will also be provided in the Id column.

Note, Insert and Update operations writing back Id and Error values to the Load table are fully supported in the SOAP API. This is similarly supported in Version 1 of the Bulk API via the WAIT method. It is not possible with Version 2 of the Bulk API, although alternative support is available via the _Return table.

All other operations (Update, Delete, Undelete) for the SOAP API and Update, Delete, Harddelete for both versions of the Bulk API fully support writing back to the Error column.

Parameter	Purpose
@Operation	Operation: Insert, Update, Upsert, Delete, Undelete, BulkAPIv1Insert, BulkAPIv1Update, BulkAPIv1Delete, BulkAPIv1Upsert, BulkAPIv1Harddelete, BulkAPIv2Insert, BulkAPIv2Update, BulkAPIv2Delete, BulkAPIv2Upsert, BulkAPIv2Harddelete Define a custom batchsize (for example 25) in this form:

Parameter	Purpose
	Insert(25), Update(25), Upsert(25), Delete(25), Undelete(25) See Upsert section for special options concerning the External Id
@Env	SQL Sales Environment Name
@TableName	SQL table you have defined and created. There are certain rules to be adhered to in the structure (mainly that an Id is present, defined as nchar(18) and an Error column, defined as nvarchar(255). The name of the table is significant as the left section of the name up to the first underscore should exactly match the Salesforce object you are performing the load against. For example table "Account_TestLoad_Insert" will be recognised by SQL Sales as being a payload for the "Account" object. Similarly "TestObject__c_TestLoad_Update will be recognised by SQL Sales as being a payload for the "TestObject__c" object. Only alphanumeric and underscore characters are supported in the table name.
@Special1	Optional, in simple usage, null Options **B** = force use of the BULK connection method (if the Environment is configured for ODBC **O** = force use of the ODBC connection method (if the Environment is configured for BULK For Bulk API V1 operations: **WAIT:SERIAL** **WAIT:PARALLEL** **BACK:SERIAL** **BACK:PARALLEL** **JOB:WAIT:SERIAL** **JOB:WAIT:PARALLEL** **JOB:BACK:SERIAL** **JOB:BACK:PARALLEL** For Bulk API V2 operations: **WAIT** **BACK** **JOB:WAIT** **JOB:BACK**

Parameter	Purpose
@Special2	Used to receive a (known) Job Id for when @Special1 is one of: For when Bulk API V1 operations was specified as one of: **JOB:WAIT:SERIAL** **JOB:WAIT:PARALLEL** **JOB:BACK:SERIAL** **JOB:BACK:PARALLEL** Or for when Bulk API V2 operations was specified as one of: **JOB:WAIT** **JOB:BACK**

This is a summary of what you can expect with regards how the Load table is supported in relation to which Method you use, which Bulk API Version and which Operation.

Method	Writes to Load table	Writes to Return table
WAIT:SERIAL (Version 1)	Yes for all operations	Yes for all operations
WAIT:PARALLEL (Version 1)	Yes for all operations	Yes for all operations
BACK:SERIAL (Version 1)	Yes for Update, Delete, Harddelete (once the JOB-BACK run has completed in Salesforce and you've followed up with a WAIT) No for Insert, Upsert	Yes for all operations (once the JOB-BACK run has completed in Salesforce and you've followed up with a WAIT)
BACK:PARALLEL (Version 1)	Yes for Update, Delete, Harddelete (once the JOB-BACK run has completed in Salesforce and you've followed up with a WAIT) No for Insert, Upsert	Yes for all operations (once the JOB-BACK run has completed in Salesforce and you've followed up with a WAIT)
JOB:WAIT:SERIAL (Version 1)	Yes for Update, Delete, Harddelete (once the JOB-BACK run has completed in Salesforce and you've followed up with a WAIT) No for Insert, Upsert	Yes for all operations
JOB:WAIT:PARALLEL (Version 1)	Yes for Update, Delete, Harddelete (once the JOB-BACK run has completed in Salesforce and	Yes for all operations

Method	Writes to Load table	Writes to Return table
	you've followed up with a WAIT) No for Insert, Upsert	
JOB:BACK:SERIAL (Version 1)	Yes for Update, Delete, Harddelete (once the JOB-BACK run has completed in Salesforce and you've followed up with a WAIT) No for Insert, Upsert	Yes for all operations
JOB:BACK:PARALLEL (Version 1)	Yes for Update, Delete, Harddelete (once the JOB-BACK run has completed in Salesforce and you've followed up with a WAIT) No for Insert, Upsert	Yes for all operations
WAIT (Version 2)	Yes for Update, Delete, Harddelete No for Insert, Upsert	Yes for all operations
BACK (Version 2)	Yes for Update, Delete, Harddelete (once the JOB-BACK run has completed in Salesforce and you've followed up with a WAIT) No for Insert, Upsert	Yes for all operations
JOB:WAIT (Version 2)	Yes for Update, Delete, Harddelete (once the JOB-BACK run has completed in Salesforce and you've followed up with a WAIT) No for Insert, Upsert	Yes for all operations
JOB:BACK (Version 2)	Yes for Update, Delete, Harddelete (once the JOB-BACK run has completed in Salesforce and you've followed up with a WAIT) No for Insert, Upsert	Yes for all operations

This is a summary of what you can expect with regards the write-back to the Error column and the newly created Ids for Insert & Upsert.

Operation	SOAP API	Bulk API V1	Bulk API V2
Insert BulkAPIv1Insert BulkAPIv2Insert	Yes	Yes (via WAIT)	No (but does write to _Return table)
Update BulkAPIv1Update BulkAPIv2Update	Yes	Yes	Yes
Delete BulkAPIv1Delete BulkAPIv2Delete	Yes	Yes	Yes
Undelete	Yes	n/a	n/a
Upsert BulkAPIv1Upsert BulkAPIv2Upsert	Yes	Yes (via WAIT)	No (but does write to _Return table)
BulkAPIv1Harddelete BulkAPIv2Harddelete	n/a	Yes	Yes

SSId

By default, SQL Sales will add a primary key of SSId, defined as an integer identity(1,1). This key is how SQL Sales processes the data at the back end and maintains data integrity as it performs the various loads. You can provide you own SSId field if that works better for your use case, however it must be defined as an integer identity(1,1). If it is not, the following error message will be thrown and the process will terminate:

```
SQL-SALES insert run date: 2023-11-04 ---------------------------
20:15:03: Using Env|Schema: DEMO|dbo
20:15:03: Provided Load Table: Account_TestLoad_Insert (for schema: dbo) contains an integer
field called SSId which is not defined as an identity column
-----------------------------------------------------------------
```

Metadata checks

It is quite normal to want or need to have additional columns in your load table, that assist in what you're trying to achieve. SQL Sales will perform a validation of each column against the available columns for the given load object, if the column either doesn't exist or can't be operated against due to the User permissions of the Username defined in the Environment configuration, then SQL Sales will simply ignore that column in the given data operation (and report back that it has been ignored).

In the example below, the Column "HelpderData" has been included in the build of the load table:

```
drop table if exists Account_TestLoad_Insert
create table Account_TestLoad_Insert
(Id nchar(18)
,Error nvarchar(255)
,Name nvarchar(255)
,HelperData nvarchar(255))

insert Account_TestLoad_Insert
(Name
,HelperData)

select 'Test1 for SQL-Sales','Info1'
union select 'Test2 for SQL-Sales','Info2'
```

The Insert will run as normal, but note the exclusion information message on the output report:

```
SQL-SALES insert run date: 2023-11-04 ---------------------------
20:17:59: Using Env|Schema: DEMO|dbo
20:17:59: Starting Loader for Account batchsize 200
20:17:59: SSId added to Account_TestLoad_Insert
20:18:02: Connection method BULK & SOAP API
20:18:02: Columns checked against Salesforce metadata
20:18:04: Excluded: HelperData is not available on object Account
20:18:04: Load complete: Success:2 Failure:0
------------------------------------------------------------------
```

Batchsize

Note by default, with the SOAP API, without specifying a batchsize, 200 is used, the maximum for the SOAP api.

Customise this by passing in the operation in the format:

<operation>(<batchsize>)

for example: Insert(25) - repeating the example:

```
exec ss_Loader 'Update(25)','DEMO','Account_TestLoad_Update'
```

As you'll discover in the later BULK API sections, you can also specify a custom batchsize (from default 10000) with the Version 1 Bulk API, for example:

```
exec ss_Loader 'BulkAPIv1Update(5000)','DEMO','Account_TestLoad_Update',’WAIT:SERIAL’
```

The Version 2 Bulk API does not support batchsize (in practical terms this means Salesforce will generally apply a batchsize of 10000).

Insert

Create a load table Example

This example creates a load table called "Account_TestLoad_Insert". The Id and Error columns are mandatory, as this is an Insert there is nothing to add into Id, but it needs to be provided, as the resultant created record Id will be passed back on creation. All load operations require the Error column, whether success or failure, to guide and inform you.

```
drop table if exists Account_TestLoad_Insert
create table Account_TestLoad_Insert
(Id nchar(18)
,Error nvarchar(255)
,Name nvarchar(255))

insert Account_TestLoad_Insert
(Name)

select 'Test1 for SQL-Sales'
union select 'Test2 for SQL-Sales'
```

Running the example

```
exec ss_Loader 'Insert','DEMO','Account_TestLoad_Insert'
```

```
SQL-SALES Insert run date: 2023-11-04 ---------------------------
18:35:30: Using Env|Schema: DEMO|dbo
18:35:30: Starting Loader for Account batchsize 200
18:35:30: SSId added to Account_TestLoad_Insert
18:35:33: Connection method BULK & SOAP API
18:35:33: Columns checked against Salesforce metadata
18:35:38: Load complete: Success:2 Failure:0
-----------------------------------------------------------------
```

As described, the created Id for the insert(s) will be passed back to the load table

```
select * from Account_TestLoad_Insert
```

sults | Messages

Id	Error	Name	SSId
0018d00000rQ5uCA...	Success	Test1 for SQL-Sales	1
0018d00000rQ5uDA...	Success	Test2 for SQL-Sales	2

Update

Create a load table Example

This example creates a load table called "Account_TestLoad_Upsert". The Id and Error columns are mandatory. All load operations require the Error column, whether success or failure, to guide and inform you. Update requires a populated Id column.

```
drop table if exists Account_TestLoad_Update
select
Id
,convert(nvarchar(255),null) as Error
,convert(bit,1) as TestCheckbox__c
-----
,TestCheckbox__c as TestCheckbox__c_Orig
-----
,'HelperTextExample' as RandomAdditionalText_Info
into Account_TestLoad_Update
from Account
```

In this example, the Update payload has been generated from the prior replicated Account table. The field "TestCheckbox__c" is currently set to 0 / FALSE. The script above has prepared an update payload, setting to 1 / TRUE. As a good practice, to preserve the original value in the payload table, we recommend you embed the original value to keep a record of the value prior to the update change.

This example also illustrates that you can have "helper" columns also in the payload.

SQL Sales best practise is to have original values included, with a "_Orig" (for original) suffix added to the column name. Similarly informational columns, useful in working with the payload should have the suffix "_Info" (for information).

To be clear, these suffix conventions are not expected or mandatory whatsoever, we just try and encourage their use, but of course, how you use SQL Sales is up to you.
Note, to get near real time accuracy, it is recommended to run a ss_Delta just before you grab the Account data (for this particular example which is sourcing from Salesforce itself).

Running the example

```
exec ss_Loader 'Update','DEMO','Account_TestLoad_Update'
```

```
SQL-SALES Update run date: 2023-11-04 ----------------------------
19:03:54: Using Env|Schema: DEMO|dbo
19:03:55: Starting Loader for Account batchsize 200
19:03:55: SSId added to Account_TestLoad_Update
19:03:58: Connection method BULK & SOAP API
19:03:58: Columns checked against Salesforce metadata
19:04:11: Excluded: RandomAdditionalText_Info is not available on object Account
19:04:11: Excluded: TestCheckbox__c_Orig is not available on object Account
19:04:11: Load complete: Success:1801 Failure:0
------------------------------------------------------------------
```

Note the reported column exclusions:
Excluded: RandomAdditionalText_Info is not available on object Account
Excluded: TestCheckbox__c_Orig is not available on object Account

```
select * from Account_TestLoad_Update
```

esults | Messages

Id	Error	TestCheckbox__c	TestCheckbox__c_Orig	RandomAdditionalText_In...	SSId
0018d00000cjGjBAAU	Success	1	0	HelperTextExample	1
0018d00000cjGvCAAU	Success	1	0	HelperTextExample	2
0018d00000cjH5MAAU	Success	1	0	HelperTextExample	3
0018d00000cjH7rAAE	Success	1	0	HelperTextExample	4

Delete

Create a load table Example

This example creates a load table called "Account_TestLoad_Delete". The Id and Error columns are mandatory. All load operations require the Error column, whether success or failure, to guide and inform you. Delete requires a populated Id column.

```
drop table if exists Account_TestLoad_Delete
select
Id
,convert(nvarchar(255),null) as Error
,Name
-----
,'HelperTextExample' as RandomAdditionalText_Info
into Account_TestLoad_Delete
from Account
```

In this example, the Delete payload has been generated from the prior replicated Account table. Additional fields (for example "Name" and "RandomAdditionalText_Info") can be included in the payload, if required or necessary for your particular use case. The Delete operation will ignore (but preserve) them as it is only concerned with the Id and Error fields.

This example also illustrates some failure scenarios.

Running the example

```
exec ss_Loader 'Delete','DEMO','Account_TestLoad_Delete'
```

```
SQL-SALES Delete run date: 2023-11-04 ----------------------------
19:23:17: Using Env|Schema: DEMO|dbo
19:23:17: Starting Loader for Account batchsize 200
19:23:17: SSId added to Account_TestLoad_Delete
19:23:20: Connection method BULK & SOAP API
19:23:20: Columns checked against Salesforce metadata
19:23:57: Load complete: Success:1791 Failure:10
-----------------------------------------------------------------
```

Note the failures being reported in this example

```
select * from Account_TestLoad_Delete
```

Messages

Id	Error	Name	RandomAdditionalText_Info	SSId
0018d00000nM9U0AAK	Success	20230920av1__0018d0...	HelperTextExample	914
0018d00000nM9U1AAK	Success	20230920av1__0018d0...	HelperTextExample	915
0018d00000nM9U2AAK	Success	20230920av1__0018d0...	HelperTextExample	916
0018d00000nM9U3AAK	Success	20230920av1__0018d0...	HelperTextExample	917
0018d00000nM9U4AAK	Success	20230920av1__0018d0...	HelperTextExample	918
0018d00000P4XctAAF	Your attempt to delete NameChange7gAAA could not be completed because it is associated with the following entitlements.: Sample E...	NameChange7gAAA	HelperTextExample	919
0018d00000P8DFzAAN	Your attempt to delete NameChange7gAAA could not be completed because it is associated with the following cases.: 00001000, 000...	NameChange7gAAA	HelperTextExample	920
0018d00000P8DG0A...	Your attempt to delete NameChange7gAAA could not be completed because it is associated with the following cases.: 00001019, 000...	NameChange7gAAA	HelperTextExample	921
0018d00000P8DG3A...	Your attempt to delete NameChange7gAAA could not be completed because it is associated with the following cases.: 00001007, 000...	NameChange7gAAA	HelperTextExample	922
0018d00000P8DG4A...	Your attempt to delete NameChange7gAAA could not be completed because it is associated with the following cases.: 00001001, 000...	NameChange7gAAA	HelperTextExample	923
0018d00000P8DG5A...	Your attempt to delete NameChange7gAAA could not be completed because it is associated with the following cases.: 00001003, 000...	NameChange7gAAA	HelperTextExample	924
0018d00000P8DG6A...	Your attempt to delete NameChange7gAAA could not be completed because it is associated with the following cases.: 00001012Your ...	NameChange7gAAA	HelperTextExample	925
0018d00000P8DG7A...	Your attempt to delete Mar 7 2023 3:20PM could not be completed because it is associated with the following cases.: 00001009	Mar 7 2023 3:20PM	HelperTextExample	926
0018d00000P8DG8A...	Your attempt to delete Mar 7 2023 3:20PM could not be completed because it is associated with the following cases.: 00001010, 0000...	Mar 7 2023 3:20PM	HelperTextExample	927
0018d00000P8DG9A...	Your attempt to delete Mar 7 2023 3:20PM could not be completed because it is associated with the following cases.: 00001006, 0000...	Mar 7 2023 3:20PM	HelperTextExample	928
0018d00000rQ5uCAAS	Success	Test1 for SQL-Sales	HelperTextExample	929
0018d00000rQ5uDAAS	Success	Test2 for SQL-Sales	HelperTextExample	930
0018d00000RzG59AAF	Success	Mar 7 2023 3:20PM	HelperTextExample	931

Here we can see the standard data provided by Salesforce in the Developer instance cannot be deleted due to referential integrity validation constraints and reported back via the Salesforce api to SQL Sales and in turn the Error column.

Undelete

Create a load table Example

This example creates a load table called "Account_TestLoad_Undelete". The Id and Error columns are mandatory. All load operations require the Error column, whether success or failure, to guide and inform you. Undelete requires a populated Id column.

```
drop table if exists Account_TestLoad_Undelete
select
Id
,convert(nvarchar(255),null) as Error
into Account_TestLoad_Undelete
from Account_TestLoad_Delete
where Error = 'Success'
```

In this example, the Undelete payload has been generated from the prior Delete payload example, taking only those which were successfully deleted. As usual, additional fields can be included in the payload, if required or necessary for your particular use case. The Delete operation will ignore (but preserve) them as it is only concerned with the Id and Error fields

Running the example

```
exec ss_Loader 'Undelete','DEMO','Account_TestLoad_Undelete'
```

```
SQL-SALES Undelete run date: 2023-11-04 -------------------------
19:34:28: Using Env|Schema: DEMO|dbo
19:34:28: Starting Loader for Account batchsize 200
19:34:28: SSId added to Account_TestLoad_Undelete
19:34:31: Connection method BULK & SOAP API
19:34:31: Columns checked against Salesforce metadata
19:35:06: Load complete: Success:1791 Failure:0
------------------------------------------------------------------
```

Upsert

Upsert with SQL Sales and working with the Salesfore api is a little different to the other operations. Update, Delete and Undelete all work from the provided Id, with regards what records to operate against. Insert merely creates new records and passes the new Id back.

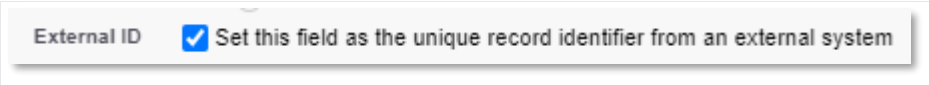
Whereas Upsert, as the name suggests, works primarily off a provided External Id for the given object being upserted against.

If a match is found in that Salesforce object for the value being passed in, then the Operation will update the provided fields in the table payload to the matched record.

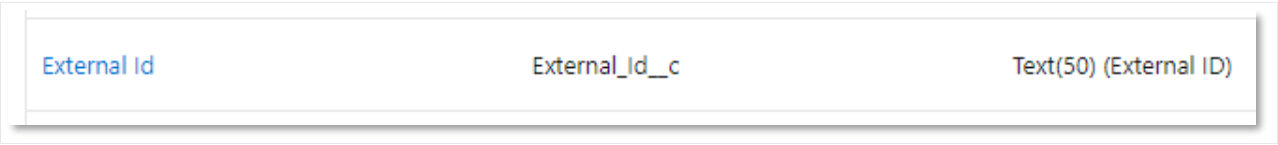
If a match is not found, the Upsert operation will insert a new record, using the provided fields in the table payload.

External Id

Upsert will only work against an External data type field. This is a special setting for a field in Salesforce:



For this example, the field "External_Id__c" has been added to the Account object, note the special indicator "(External ID)". If you intended External Id does not have this, it is likely not actually setup as an External Id, no matter what the field name is.



SQL Sales will inform if it is not truly an External Id, this is demonstrated in the following examples.

Create a load table Example (Prep)

This example creates a load table called "Account_TestLoad_Update", to populate the newly created External_Id__c with values, this is merely background preparation so this example can work off values in Salesforce. Remember, for Updates, the Id and Error columns are mandatory. All load operations require the Error column, whether success or failure, to guide and inform you. Update requires a populated Id column.

```
drop table if exists Account_TestLoad_Update
select
Id
,convert(nvarchar(255),null) as Error
,AccountNumber as External_Id__c
,External_Id__c as External_Id__c_Orig
into Account_TestLoad_Update
from Account
where AccountNumber is not null
```

Running the example (prep)

This is being provided as an educational guide to using ss_Loader and is not directly relevant for Upsert, the specific Upsert instructions are coming up in the next section

```
exec ss_Loader 'Update','DEMO','Account_TestLoad_Update'
```

Main Example

Examining the prepared Upsert payload, note that there is a column called "XId". This is the column which holds the value to be matched against the External Id specific to your use case (see detail coming up). Note also that three fields have values in XId, in fact two of these exist in Salesforce in field Account.External_Id__c, these will be matched and hence updated. The third value "XXXYYYZZZ" does not exist and so will be inserted. The fourth record has no value in XId and so will not be matched either, resulting in a failure as the External Id must be provided.

```
select * from Account_TestLoad_Upsert

select External_Id__c,Name
from Account where External_Id__c in ('CD451796','CD656092')
```

esults Messages

Id	Error	XId	Name
NULL	NU...	NULL	Expected failure as no value provided in XId
NULL	NU...	CD451796	New Name, Upsert Demo 123
NULL	NU...	CD656092	New Name, Upsert Demo 789
NULL	NU...	XYZ123456789	Test Insert1 as no XId match

External_Id__c	Name
CD451796	NameChange7gAAA
CD656092	NameChange7gAAA

Running the example

Note, the XId column is assigned to the actual External api field name with the switch convention:

```
Upsert:XId=External_Id__c
```

```
exec ss_Loader 'Upsert:XId=External_Id__c','DEMO','Account_TestLoad_Upsert'
```

```
SQL-SALES Upsert:XId=External_Id__c run date: 2023-11-04 --------
19:59:24: Using Env|Schema: DEMO|dbo
19:59:26: Starting Loader for Account batchsize 200
19:59:26: SSId added to Account_TestLoad_Upsert
19:59:29: Connection method BULK & SOAP API
19:59:29: Columns checked against Salesforce metadata
19:59:32: Load complete: Success:3 Failure:1
------------------------------------------------------------------
```

Here we can observe from the load table that as expected, three records are successful and the fourth has failed as no External Id has been provided.

```
select * from Account_TestLoad_Upsert
```

esults Messages

Id	Error	XId	Name	SSId
	External_Id__c not specified	NULL	Expected failure as no value provided in XId	1
0018d00000P8DFzAAN	Success	CD451796	New Name, Upsert Demo 123	2
0018d00000P8DG0A...	Success	CD656092	New Name, Upsert Demo 789	3
0018d00000rQ6QCAA0	Success	XYZ123456789	Test Insert1 as no XId match	4

The next test will be to run ss_Delta, where we expect 2 Updates and 1 Insert:

```
exec ss_Delta 'DEMO', 'Account'
```

```
SQL-SALES Delta run date: 2023-11-04 ----------------------------
20:03:53: Using Env|Schema: DEMO|dbo
20:03:53: Starting Delta Replication, checking input Account
20:03:58: Provided Objectname Account is valid with batchsize 2000
20:03:58: Delta schema successfully built using connection method BULK & SOAP API
20:03:58: Data population fully completed
20:03:58: 3 rows delta replicated: Insert:1 Update:2 Delete:0
-----------------------------------------------------------------
```

Querying the delta refreshed local Account table, we can see the pre-existing and matched-to records have been updated with the "__StringAdded" text appended to the Name field, whereas the unmatched third record has been inserted (created).

```
select External_Id__c,Name
from Account where External_Id__c in ('CD451796','CD656092','XYZ123456789')
```

Results | Messages

External_Id__c	Name
CD451796	New Name, Upsert Demo 123
CD656092	New Name, Upsert Demo 789
XYZ123456789	Test Insert1 as no Xld match

Invalid input

The field name passed in must be both a valid Salesforce field and crucially one which is defined as an External Id.

The below attempts are a field name that does not exist in Salesforce followed by a field name that does exist but which is not defined as an External Id

```
exec ss_Loader 'Upsert:XId=ExternalId','DEMO','Account_TestLoad_Upsert'
```

```
SQL-SALES Upsert:XId=ExternalId run date: 2023-11-04 ------------
20:06:43: Using Env|Schema: DEMO|dbo
20:06:46: Provided External Id: "ExternalId" for the Upsert operation
20:06:46: is not defined as an External Id on the Account object
-----------------------------------------------------------------
```

The final check is that the expected formatting convention of "Upsert:XId=<External field name>" for example "Upsert:XId=External_Id__c" does rely on the ":XId=" section being passed in correctly, below is an example where that has not been passed in as expected:

```
exec ss_Loader 'Upsert:X=External_Id__c','DEMO','Account_TestLoad_Upsert'
```

```
SQL-SALES Upsert:X=External_Id__c run date: 2023-11-04 ----------
20:10:02: Using Env|Schema: DEMO|dbo
20:10:02: Provided Upsert Operation value not in the correct format, for a hypothetical field
called "External_Id__c"
20:10:02: set (in Salesforce) as an External Id (this is a field setting)
20:10:02: the expected input value for the standard web services API is:
Upsert:XId=External_Id__c
20:10:02: or BulkAPIUpsert:XId=External_Id__c for the bulk API
20:10:02: with a defined batchsize (for example 100 or 1000 respectively) these would be:
20:10:02: Upsert(100):XId=External_Id__c | BulkAPIv1Upsert(10000):XId=External_Id__c |
BulkAPIv2Upsert(2000):XId=External_Id__c
20:10:02: Note, the provided External Id is validated directly against Salesforce Account
20:10:02: prior to the run commencing to check it is actually defined as an External Id.
-----------------------------------------------------------------
```

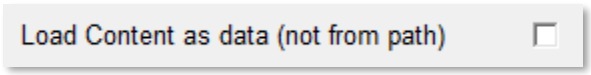
Files and Notes

Working with the Salesforce Content model can be challenging. SQL Sales offers multiple ways to work with the model, based on real world experiences of moving note data and files around.

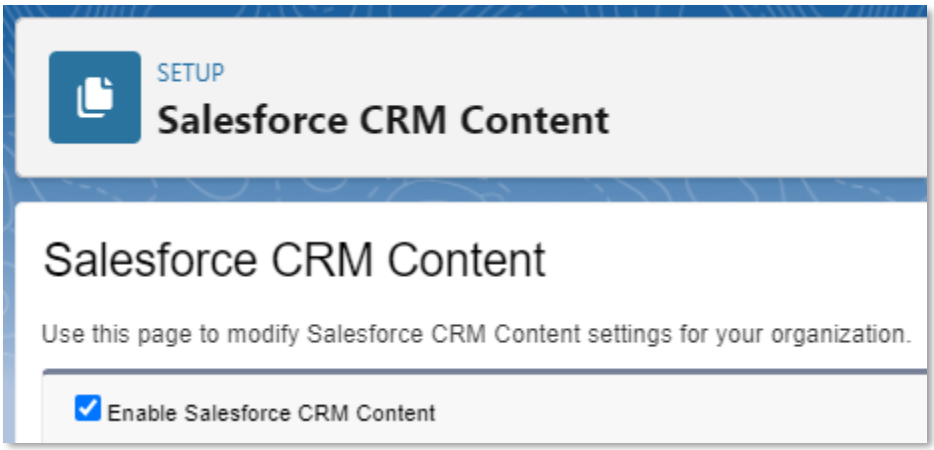
ContentNote (from a file)

The standard way to load a ContactNote record is to associate the load payload insert record with a file, residing in a UNC path, accessible to SQL Sales.

By default, from installation, SQL Sales assumes you will be loading via the standard api approach of passing in a location of a file. Ensure the configuration of your given Environment is as below:

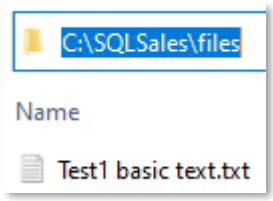


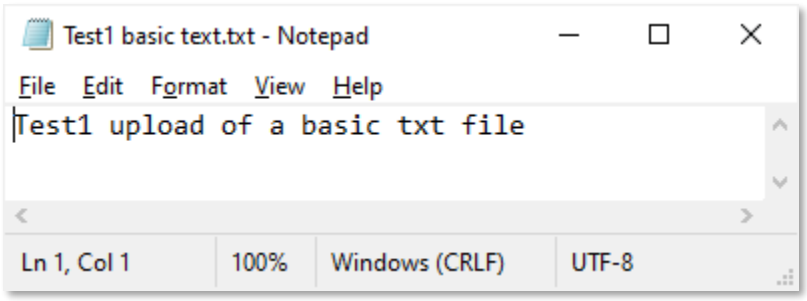
Note, additionally ensure you have enabled the Salesforce CRM Content setting (see below)



Basic text example

In this example a basic .txt file, as seen below in notepad and in the path shown, will be loaded to Salesforce as a ContentNote.





Prepare the payload

```
drop table if exists ContentNote_Test1_Insert
select
convert(nchar(18),null) as Id
,convert(nvarchar(255),null) as Error
,convert(nvarchar(255),'Test1 Title') as Title
,convert(nvarchar(max),'C:\SQLSales\files\Test1 basic text.txt') as Content
into ContentNote_Test1_Insert
```

Running the example

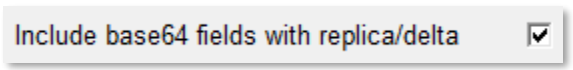
```
exec ss_Loader 'insert','demo','ContentNote_Test1_Insert'
```

```
SQL-SALES insert run date: 2023-11-11 ----------------------------
16:19:42: Using Env|Schema: demo|dbo
16:19:42: Starting Loader for ContentNote batchsize 200
16:19:42: SSId added to ContentNote_Test1_Insert
16:19:46: Connection method BULK & SOAP API
16:19:46: Columns checked against Salesforce metadata
16:19:48: Load complete: Success:1 Failure:0
-----------------------------------------------------------------
```

Examine the output

You are recommended to return the test example, to familiarise yourself with the Salesforce Content data model as we work through these examples

Modify the below configuration setting depending on whether you want to return fields such as ContentNote.Content and ContentVersion.VersionData. For large volumes records, it is not always necessary to hold the binary data in the base64 fields, hence a more efficient approach to managing the replication of objects like this, is to bypass by unchecking this setting.




```
exec ss_Replica 'demo', 'ContentNote'
```

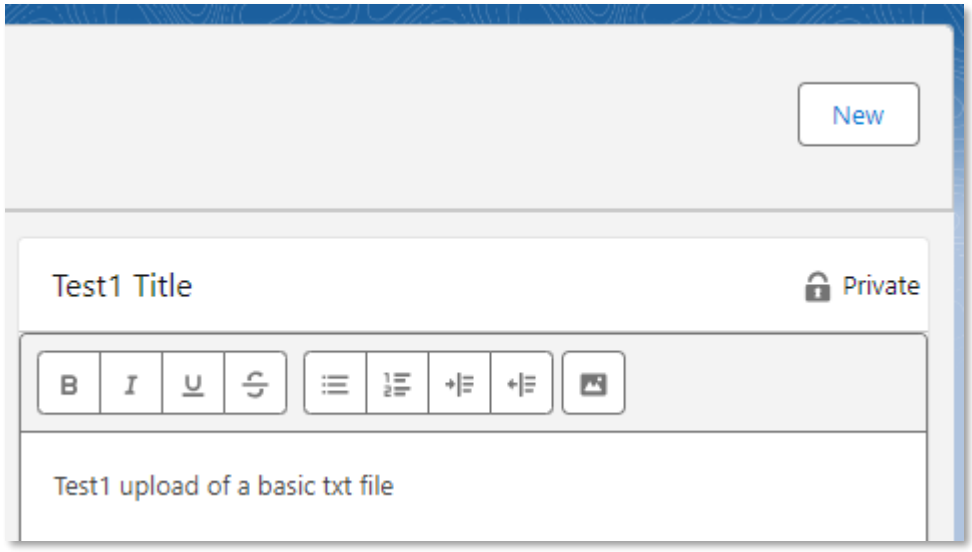
```
exec ss_Replica 'demo', 'ContentVersion'
exec ss_Replica 'demo', 'ContentDocument'
```

By inserting this one ContentNote record, Salesforce has created the following:

- ContentNote holds the Note, converted to a file in Salesforce, the binary data of which is held in ContentNote.Content.
- ContentNote.Id is in fact the ContentDocument.Id
- ContentNote.LatestPublishedVersionId is the ContentVersion.Id
- ContentVersion.FirstPublishLocationId is by default the Load User typically or ContentNote.OwnerId if you had specified one
- ContentNote.Content is the same as ContentVersion.VersionData

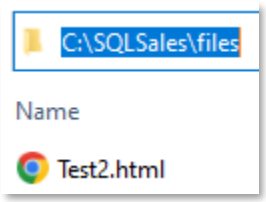
ContentNote is in effect an extension of the ContentDocument itself (as ContentNote.Id is the ContentDocument.Id).

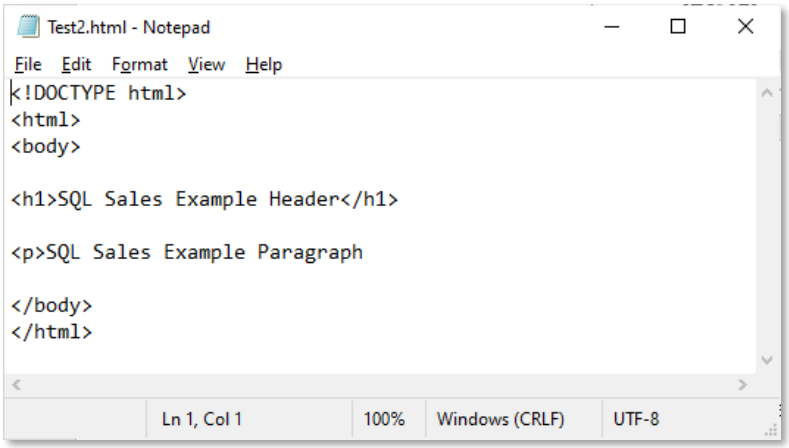
Shown in Salesforce:



HTML text example

In this example an html txt file, as seen below in notepad and opening in a browser, will be submitted. Salesforce will happily load an html formatted document to a ContentNote record. The path for this example is also shown below.





Prepare the payload

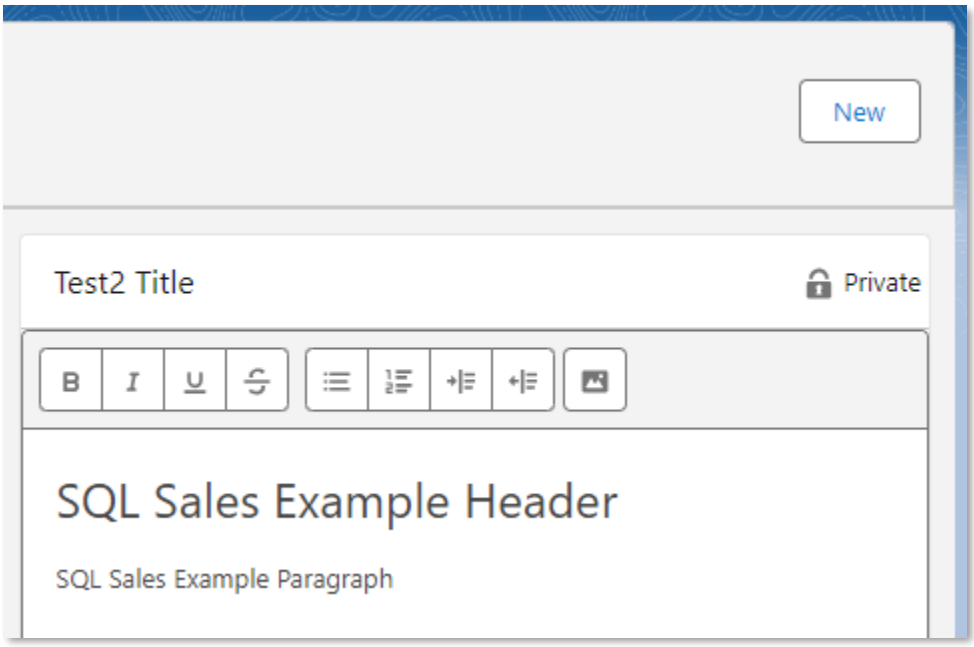
```
drop table if exists ContentNote_Test2_Insert
select
convert(nchar(18),null) as Id
,convert(nvarchar(255),null) as Error
,convert(nvarchar(255),'Test2 Title') as Title
,convert(nvarchar(max),'C:\SQLSales\files\Test2.html') as Content
into ContentNote_Test2_Insert
```

Running the example

```
exec ss_Loader 'insert','demo','ContentNote_Test2_Insert'
```

```
SQL-SALES insert run date: 2023-11-11 ---------------------------
17:34:58: Using Env|Schema: demo|dbo
17:34:58: Starting Loader for ContentNote batchsize 200
17:34:58: SSId added to ContentNote_Test2_Insert
17:35:01: Connection method BULK & SOAP API
17:35:01: Columns checked against Salesforce metadata
17:35:04: Load complete: Success:1 Failure:0
-----------------------------------------------------------------
```

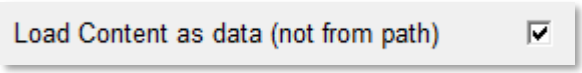
Shown in Salesforce:



ContentNote (from data), basic text

The alternative method which SQL Sales has made possible, is to load SQL Server data to ContentNote, without the need for a file of the Note to be referenced as with the previous approach.

You will have to ensure the configuration of your given Environment is as below (as the data will be loaded as data and not from a file):



Basic text example

In this example, the hard-coded text "Test, direct text approach" is representing a varchar / nvarchar text field you may have from a data source you have staging in SQL Server. Note that the preparation above converts to data type varbinary(max).

```
drop table if exists ContentNote_Test3_Insert
select
convert(nchar(18),null) as Id
,convert(nvarchar(255),null) as Error
,convert(nvarchar(255),'Test3 Title') as Title
,convert(varbinary(max),'Test, direct text approach') as ContentText
,convert(varchar(max),null) as Content
into ContentNote_Test3_Insert
```

The next step is to convert, using the above code, the varbinary(max) to varchar(max). Use these code snippets as a foundation for your own TSQL coding, likely with many more records.

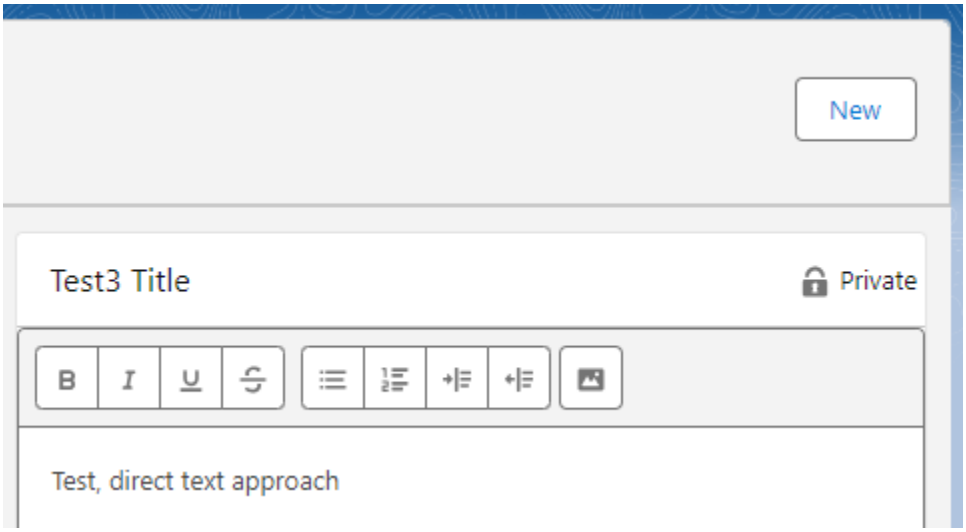
```
update ContentNote_Test3_Insert
set Content = convert(varchar(max),CAST(N'' AS
XML).value('xs:base64Binary(xs:hexBinary(sql:column("ContentText")))', 'varchar(max)'))
```

Running the example

```
exec ss_Loader 'insert','demo','ContentNote_Test3_Insert'
```

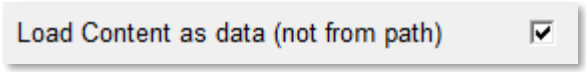
```
SQL-SALES insert run date: 2023-11-11 ----------------------------
18:46:22: Using Env|Schema: demo|dbo
18:46:22: Starting Loader for ContentNote batchsize 200
18:46:22: SSId added to ContentNote_Test3_Insert
18:46:27: Connection method BULK & SOAP API
18:46:27: Columns checked against Salesforce metadata
18:46:30: Excluded: ContentText is not available on object ContentNote
18:46:30: Load complete: Success:1 Failure:0
-----------------------------------------------------------------
```

Shown in Salesforce:



ContentNote (from data), existing base64

As with the previous basic example, you will have to ensure the configuration of your given Environment is as below (as the data will be loaded as data and not from a file):



In this example, we will be taking the existing base64 binary data directly from SQL Server, in this case from the ContentNote.Content varbinary(max) data from the previously loaded examples.

Existing base64 example

```
drop table if exists ContentNote_Test4_Insert
select
convert(nchar(18),null) as Id
,convert(nvarchar(255),null) as Error
,convert(nvarchar(255),'Test4 Title (from ' + Title + ')') as Title,convert(varchar(max),CAST(N''
AS XML).value('xs:base64Binary(xs:hexBinary(sql:column("Content")))', 'varchar(max)')) as Content
into ContentNote_Test4_Insert
from ContentNote
```

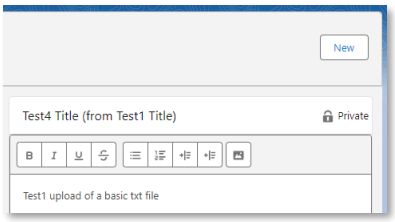
Running the example

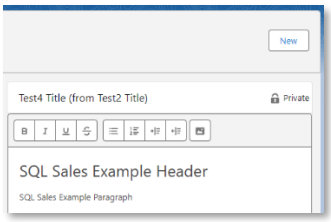
```
exec ss_Loader 'insert','demo','ContentNote_Test4_Insert'
```

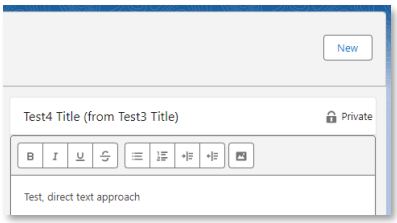
```
SQL-SALES insert run date: 2023-11-11 ----------------------------
19:07:06: Using Env|Schema: demo|dbo
19:07:06: Starting Loader for ContentNote batchsize 200
19:07:06: SSId added to ContentNote_Test4_Insert
```

```
19:07:08: Connection method BULK & SOAP API
19:07:08: Columns checked against Salesforce metadata
19:07:12: Load complete: Success:3 Failure:0
---------------------------------------------------------------
```

Shown in Salesforce:



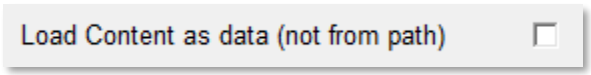




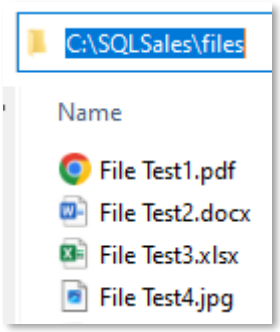
ContentVersion (from a file to a file held in Salesforce)

The standard way to load a ContentDocument (ContentVersion) record is to associate the load payload insert record with a file, residing in a UNC path, accessible to SQL Sales and therefore also accessible to the SQL Server service account which is running your MSSQL.

By default, from installation, SQL Sales assumes you will be loading via the standard api approach of passing in a location of a file. Ensure the configuration of your given Environment is as below:



Any file can be loaded to Salesforce, in the following example, four files of different types will be loaded in the same payload.



Prepare the payload

Your ContentVersion payload table must contain these fields

Field	Datatype	Purpose
Id	nchar(18)	null on the insert payload
Error	nvarchar(255)	null on the insert payload
Title	nvarchar(255)	Title of the loaded document
Origin	char(1)	C = (standard option) for Content H = Salesforce files from the user's My Files (Chatter but also for files external to Salesforce)
ContentLocation	char(1)	S = Document is in Salesforce E = Document is outside of Salesforce L = Document is on a Social Network
Description	nvarchar(255)	Description of the loaded document
VersionData	nvarchar(500)	Full path to the file, including the filename
PathOnClient	nvarchar(255)	The filename
ContentUrl	nvarchar(255)	Not Required for this example
FirstPublishLocationId	nchar(18)	If left blank this will automatically be set as the load User's User.Id, otherwise you can specify a User or Object (i.e. the "Parent" record, for example Account). Alternatively, you can specify a Library (a ContentWorkspace).

In this example, we will set the FirstPublishLocationId as an Account record, this will automatically set the "Parent" as this Account record, i.e. a ContentDocumentLink record will be automatically created with the LinkedEntityId set as the Account.Id.

Additionally in this example, we could set the ContentVersion.OwnerId as a User other than the load user. This will automatically create a second ContentDocumentLink record where the LinkedEntityId is set as the User.Id used in the ContentVersion.OwnerId.

If no ContentVersion.OwnerId is set, the default will be set as the load user (i.e. the username defined in the given Environment). If the load user is not set as the ContentVersion.OwnerId then it will be more challenging retrieving (replicating) the ContentVersion records just inserted, as the load user will not have sufficient permissions (as no entry in ContentDocumentLink will exist).

Example

```
drop table if exists ContentVersion_Insert
create table ContentVersion_Insert
(Id nchar(18)
```

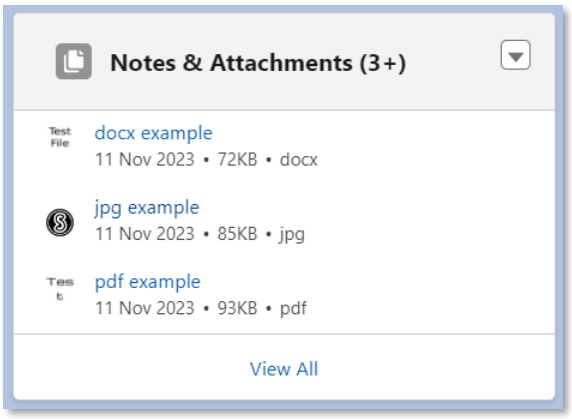
```
,Error nvarchar(255)
,Title nvarchar(255)
,ContentDocumentId nchar(18)
,Origin char(1)
,ContentLocation char(1)
,OwnerId nchar(18)
,Description nvarchar(255)
,VersionData nvarchar(500)
,PathOnClient nvarchar(255)
,FirstPublishLocationId nchar(18))

insert ContentVersion_Insert
(Title
,Origin
,ContentLocation
,Description
,VersionData
,PathOnClient
,FirstPublishLocationId)

select
'pdf example' --Title
,'C' --Origin
,'S' --ContentLocation
,'Example of loading a pdf from a file' --Description
,'C:\SQLSales\files\File Test1.pdf' --VersionData
,'File Test1.pdf' --PathOnClient
,'0018d00000cjHAvAAM' --FirstPublishLocationId

union select
'docx example' --Title
,'C' --Origin
,'S' --ContentLocation
,'Example of loading a docx from a file' --Description
,'C:\SQLSales\files\File Test2.docx' --VersionData
,'File Test2.docx' --PathOnClient
,'0018d00000cjHAvAAM' --FirstPublishLocationId

union select
'xlsx example' --Title
,'C' --Origin
,'S' --ContentLocation
,'Example of loading a xlsx from a file' --Description
,'C:\SQLSales\files\File Test3.xlsx' --VersionData
,'File Test3.xlsx' --PathOnClient
,'0018d00000cjHAvAAM' --FirstPublishLocationId

union select
'jpg example' --Title
,'C' --Origin
,'S' --ContentLocation
,'Example of loading a jpg from a file' --Description
,'C:\SQLSales\files\File Test4.jpg' --VersionData
,'File Test4.jpg' --PathOnClient
,'0018d00000cjHAvAAM' -FirstPublishLocationId
```

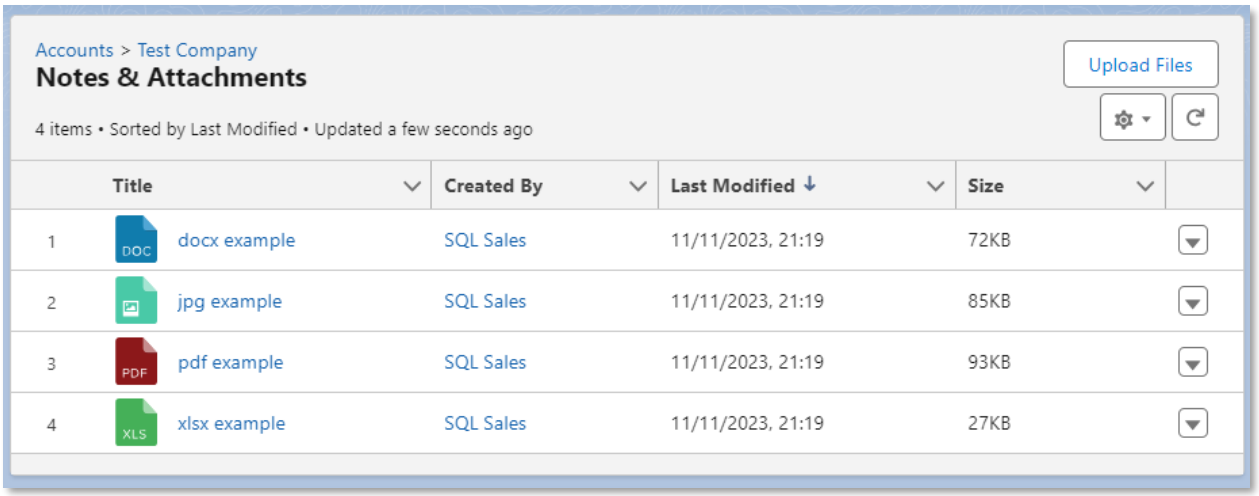
Running the example

```
exec ss_Loader 'insert','demo','ContentVersion_Test1_Insert'
```

```
SQL-SALES insert run date: 2023-11-11 ---------------------------
21:19:39: Using Env|Schema: demo|dbo
21:19:39: Starting Loader for ContentVersion batchsize 200
21:19:39: SSId added to ContentVersion_Test1_Insert
21:19:42: Connection method BULK & SOAP API
21:19:42: Columns checked against Salesforce metadata
21:19:47: Load complete: Success:4 Failure:0
-----------------------------------------------------------------
```

Shown in Salesforce:

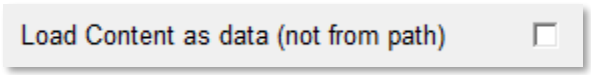




ContentVersion (External link only, not held in Salesforce)

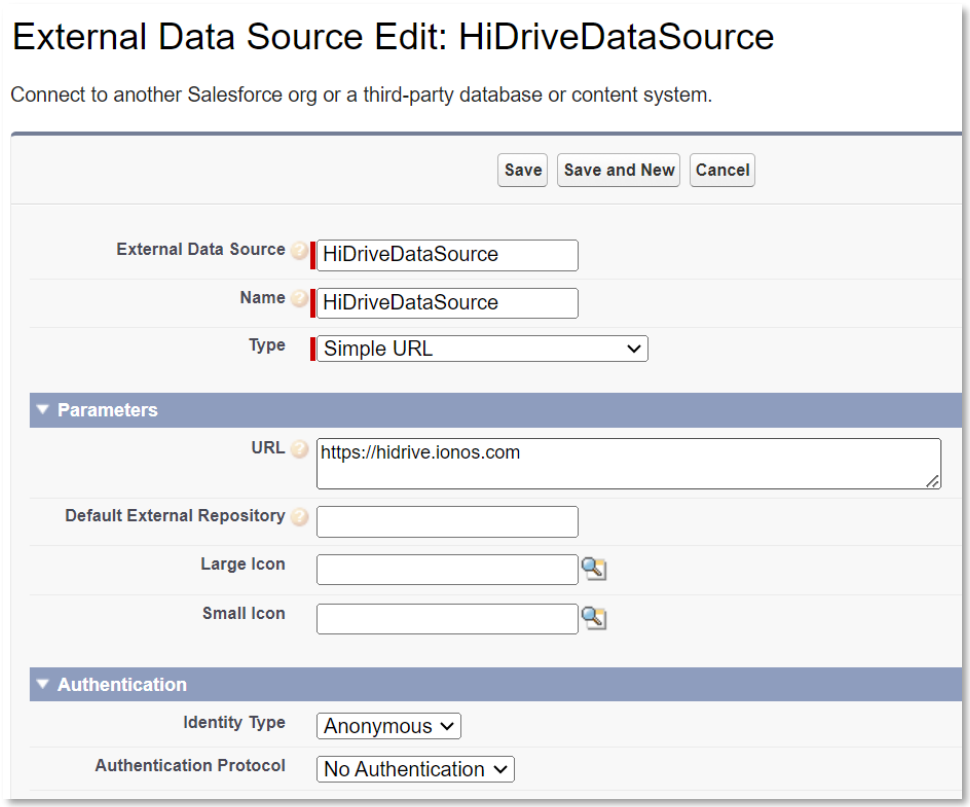
An alternative approach is to load only links to Content held outside of Salesforce.

As with the previous standard approach, by default, from installation, SQL Sales assumes you will be loading via the standard api approach of passing in a location of a file. Ensure the configuration of your given Environment is as below:



In the example to follow, there is no authentication required to access the external file however an ExternalDataSource will still need to be defined in our development Org. In practise you may need to setup more involved security for your use case, which is covered by Salesforce documentation. "HiDrive" is just a third party host provider that SQL-Sales uses but this could be any Content host external to Salesforce.

The following screen shows an ExternalDataSource being added in Salesforce setup.



This precursor step is necessary as the resultant ExternalDataSource.Id is required in the ContentVersion payload to follow. Once you have added in Salesforce, replicate back to SQL with:

```
exec ss_Replica 'demo', 'ExternalDataSource'
```

Your ContentVersion payload table must contain these fields

Field	Datatype	Purpose
Id	nchar(18)	null on the insert payload
Error	nvarchar(255)	null on the insert payload
Title	nvarchar(255)	Title of the loaded document
Origin	char(1)	H = Chatter, but also for outside of Salesforce
ContentLocation	char(1)	E = Document is outside of Salesforce
Description	nvarchar(255)	Description of the loaded document
ContentUrl	nvarchar(255)	Full url to the Content
ExternalDocumentInfo1	nvarchar(255)	Full url to the Content
ExternalDataSourceId	nchar(18)	(ExternalDataSource.Id from the earlier setup in Salesforce)
FirstPublishLocationId	nchar(18)	If left blank this will automatically be set as the load User's User.Id, otherwise you can specify a User or Object (i.e. the "Parent" record, for example Account). Alternatively, you can specify a Library (a ContentWorkspace).

In this example, we will set the FirstPublishLocationId as an Account record, this will automatically set the “Parent” as this Account record, i.e. a ContentDocumentLink record will be automatically created with the LinkedEntityId set as the Account.Id.

Additionally in this example, we could set the ContentVersion.OwnerId as a User other than the load user. This will automatically create a second ContentDocumentLink record where the LinkedEntityId is set as the User.Id used in the ContentVersion.OwnerId.

If no ContentVersion.OwnerId is set, the default will be set as the load user (i.e. the username defined in the given Environment). If the load user is not set as the ContentVersion.OwnerId then it will be more challenging retrieving (replicating) the ContentVersion records just inserted, as the load user will not have sufficient permissions (as no entry in ContentDocumentLink will exist).

Example

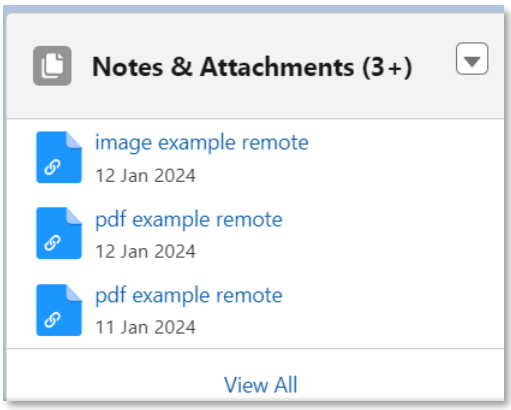
```
drop table if exists ContentVersion_ExternalTest_Insert
create table ContentVersion_ExternalTest_Insert
(Id nchar(18)
,Error nvarchar(255)
,Title nvarchar(255)
,ContentDocumentId nchar(18)
,Origin char(1)
,ContentLocation char(1)
,OwnerId nchar(18)
,Description nvarchar(255)
,ContentUrl nvarchar(255)
,ExternalDocumentInfo1 nvarchar(255)
,ExternalDataSourceId nvarchar(255)
,FirstPublishLocationId nchar(18))

insert ContentVersion_ExternalTest_Insert
(Title
,Origin
,ContentLocation
,Description
,ContentUrl
,ExternalDocumentInfo1
,ExternalDataSourceId
,FirstPublishLocationId)

select
'image example remote' --Title
,'H' --Origin
,'E' --ContentLocation
,'Example of linking to external Content' --Description
,'https://hidrive.ionos.com/lnk/JQGN0ptz' as ContentUrl
,'https://hidrive.ionos.com/lnk/JQGN0ptz' as ExternalDocumentInfo1
,'0XC8d0000000Gi0GAE' as ExternalDataSourceId
,'0018d00000P4XctAAF' FirstPublishLocationId
```

Running the example

```
exec ss_Loader 'insert','demo','ContentVersion_ExternalTest_Insert'
```

```
SQL-SALES insert run date: 2024-01-12 ---------------------------
23:20:07: Using Env|Schema: demo|dbo
23:20:07: Starting Loader for ContentVersion batchsize 200
23:20:07: SSId added to ContentVersion_ExternalTest_Insert
23:20:10: Connection method BULK & SOAP API
23:20:10: Columns checked against Salesforce metadata
23:20:10: Starting load for ContentVersion_ExternalTest_Insert
23:20:13: Load complete: Success:1 Failure:0
-----------------------------------------------------------------
```

Shown in Salesforce:



Working with ContentDocumentLink

Earlier, it was mentioned there will be permissions issues accessing ContentVersion data if the load user is not present in ContentDocumentLink for a given ContentDocument that is required (when pulling via ss_Replica or ss_Delta).

```
exec ss_Replica 'demo', 'ContentDocumentLink','LinkedEntityId','0018d00000cjHAvAAM'
select * from ContentDocumentLink
```

It is recommended to read up on the ContentDocumentLink section in this guide. As a shortcut, here we have passed in the AccountId used in this example. This returns the 4 ContentDocumentIds for the four loaded files.

ContentDocumentId	Id	IsDeleted	LinkedEntityId	ShareType	SystemModstamp	Visibility
0698d00000PZ4zYAAT	06A8d00000ecvG8EAI	0	0018d00000cjHAvA...	V	2023-11-11 21:19:45.00000...	AllUsers
0698d00000PZ4zZAAT	06A8d00000ecvGAEAY	0	0018d00000cjHAvA...	V	2023-11-11 21:19:45.00000...	AllUsers
0698d00000PZ4zaAAD	06A8d00000ecvGCEAY	0	0018d00000cjHAvA...	V	2023-11-11 21:19:45.00000...	AllUsers
0698d00000PZ4zbAAD	06A8d00000ecvGEEAY	0	0018d00000cjHAvA...	V	2023-11-11 21:19:45.00000...	AllUsers

```
exec ss_Replica 'demo',
'ContentDocumentLink','ContentDocumentId','0698d00000PZ4zYAAT,0698d00000PZ4zZAAT,0698d00000PZ4zaA
AD,0698d00000PZ4zbAAD'
select * from ContentDocumentLink
```

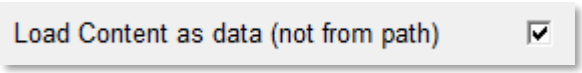
Here we see all linked entities for the ContentDocumentIds passed in.

ContentDocumentId	Id	IsDeleted	LinkedEntityId	ShareType	SystemModstamp	Visibility
0698d00000PZ4zYAAT	06A8d00000ecvG7EAI	0	0058d0000054YNTAA2	I	2023-11-11 21:19:45.00000...	AllUsers
0698d00000PZ4zYAAT	06A8d00000ecvG8EAI	0	0018d00000cjHAvAAM	V	2023-11-11 21:19:45.00000...	AllUsers
0698d00000PZ4zZAAT	06A8d00000ecvG9EAI	0	0058d0000054YNTAA2	I	2023-11-11 21:19:45.00000...	AllUsers
0698d00000PZ4zZAAT	06A8d00000ecvGAEAY	0	0018d00000cjHAvAAM	V	2023-11-11 21:19:45.00000...	AllUsers
0698d00000PZ4zaAAD	06A8d00000ecvGBEAY	0	0058d0000054YNTAA2	I	2023-11-11 21:19:45.00000...	AllUsers
0698d00000PZ4zaAAD	06A8d00000ecvGCEAY	0	0018d00000cjHAvAAM	V	2023-11-11 21:19:45.00000...	AllUsers
0698d00000PZ4zbAAD	06A8d00000ecvGDEAY	0	0058d0000054YNTAA2	I	2023-11-11 21:19:45.00000...	AllUsers
0698d00000PZ4zbAAD	06A8d00000ecvGEEAY	0	0018d00000cjHAvAAM	V	2023-11-11 21:19:45.00000...	AllUsers

ContentVersion (from data to a file held in Salesforce)

The alternative method which SQL Sales has made possible, is to load SQL Server data to ContentVersion, without the need for a file of the document to be referenced as with the previous approach.

You will have to ensure the configuration of your given Environment is as below (as the data will be loaded as data and not from a file):



In this example, we will be taking the existing base64 binary data directly from SQL Server, in this case from the ContentVersion.VersionData varbinary(max) data from the previously loaded examples.

Existing base64 example

```
drop table if exists ContentVersion_Test2_Insert
select
convert(nchar(18),null) as Id
,convert(nvarchar(255),null) as Error
,convert(nvarchar(255),'File Test2 Title (from ' + Title + ')') as Title
,PathOnClient
,convert(varchar(max),CAST(N'' AS
XML).value('xs:base64Binary(xs:hexBinary(sql:column("VersionData")))', 'varchar(max)')) as
VersionData
into ContentVersion_Test2_Insert
from ContentVersion
where Description like 'Example of loading a % from a file'
```

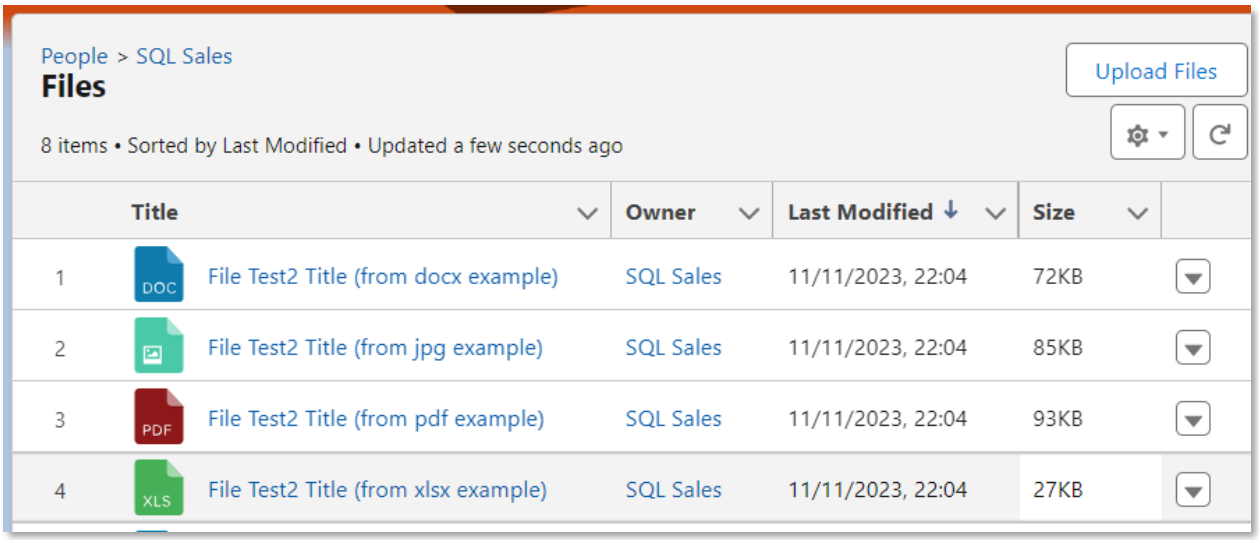
Note, PathOnClient is mandatory for VersionData, hence included in this example

Running the example

```
exec ss_Loader 'insert','demo','ContentVersion_Test2_Insert'
```

```
SQL-SALES insert run date: 2023-11-11 ----------------------------
22:04:15: Using Env|Schema: demo|dbo
22:04:15: Starting Loader for ContentVersion batchsize 200
22:04:15: SSId added to ContentVersion_Test2_Insert
22:04:18: Connection method BULK & SOAP API
22:04:18: Columns checked against Salesforce metadata
22:04:22: Load complete: Success:4 Failure:0
-----------------------------------------------------------------
```

Shown in Salesforce:



Bulk API

The Salesforce Bulk API allows batches of data to be submitted to Salesforce for processing asynchronously to the running of the ss_Loader stored procedure (i.e. once the data payload is delivered you do not need to concern yourself with the immediate stored procedure process completing. There are pros and cons to this api, vs the standard web services SOAP api (which runs synchronously to your ss_Loader stored procedure).

The main advantage of the Bulk API is for handling large volumes of data that you'd prefer to pass up to Salesforce and leave for Salesforce to process. The main disadvantage is that the overhead associated with checking the load is not as seamless as the SOAP API (i.e. the operations Insert, Update, Delete, Undelete, Upsert which have been considered thus far are all via the SOAP API). The following sections are going to consider the same operations.

Note, the SOAP API does not support Harddelete, conversely the Bulk API does not support Undelete

Version 1 of the Bulk API is the original version and still fully supported by Salesforce. The main different to the later Version 2 is that whilst a specified batchsize as such is not supported in v1, SQL Sales can control the size of each submitted batch (as V1 allows this specification), thereby in effect allowing a batchsize to be defined. Version 1 also allows a concurrency mode to be specified (i.e. either Serial or Parallel). Some use cases of submitting data against objects that have a lot of associated code, may encounter a higher chance of failure (Salesforce apex code, triggers, workflows, governor limits etc, nothing to do with SQL Sales), for such objects it can be safer to submit serially rather than in parallel, but each use case needs to be judged on its own merits. Version 1 is also more predictable in terms of how batches are created, which allows (for Inserts) SQL Sales to tie a response batch of new Ids back to the submitted data (in a similar way to the SOAP API Insert operation), provided the "WAIT" switch is employed.

Version 2 is more "automatic" in the sense you cannot specific how to define the contents of a given batch (Salesforce will decide if this will be spit across multiple batches or contained within one batch), as such batchsize is not supported with the Bulk API Version 2. Similarly, concurrency mode (Serial/Parallel) is not controllable as Salesforce again defines how a given batch or set of batches will be processed. Finally, Version 2 has no set way of predicting how a batch of responses will be returned vs how they were submitted, hence Bulk API Inserts cannot be tied back to the submitted data payload, even with the WAIT switch. This will therefore have to be done by the User, keying the

payload by for example an embedded key field that is part of the given target Salesforce object field definition. For example if you have a field “CustomId__c” on the given object and you pass in an appropriate custom Id as part of your BulkAPIv2Insert, when replicating the data back (via ss_Replica or ss_Delta) you can analyse how your Bulk API V2 has behaved.

These key differences will be reflected in the subtle switch differences in the following Version 1 vs Version 2 documented operations.

BulkAPIv1Insert

Create a load table Example (WAIT)

This example creates a load table called "Account_BulkAPIv1_Wait_Serial_Example_Insert". The Id and Error columns are mandatory, as this is an Insert there is nothing to add into Id, but it needs to be provided, as the resultant created record Id will be passed back on creation. All load operations require the Error column, whether success or failure, to guide and inform you.

WAIT means the SQL process will remain live while it waits for the Bulk API to return all responses. As the batches are created in v1 predictably, the responses can be tied back to the initial Insert payload and hence behave in a similar way to the SOAP API, in other words, the success/failure Error outcome will be posted back to the load table's Error column.

SERIAL is how the Bulk API will internally process the created batches, Serial means one at a time which is a least stressful way to submit the batches (not that this should actually make a difference, however in some edge cases, particularly on complex objects with lots of config and code, submitting in parallel mode can cause threshold/governor limit failures).

PARALLEL is how the Bulk API will internally process the created batches, Parallel means that SQL Sales will submit as many batches to Salesforce as necessary according to the provided batchsize specified. Salesforce will process the batches potentially all in parallel, although in reality they should be processed as quickly as they can (still likely at least partially in parallel) according to the governor limits defined in your Salesforce instance.

```
drop table if exists Account_BulkAPIv1_Wait_Serial_Insert
select top 100
convert(nchar(18),null) as Id
,convert(nvarchar(255),null) as Error
,'DEMO__PREFIX' + Name as Name
-----
,Name as Name_Info
,AccountNumber as AccountNumber_Info
into Account_BulkAPIv1_Wait_Serial_Insert
from Account
```

Running the example

(Pass WAIT:PARALLEL for Parallel requirements)

```
exec ss_Loader 'BulkAPIv1Insert', 'DEMO', 'Account_BulkAPIv1_Wait_Serial_Insert','WAIT:SERIAL'
```

```
SQL-SALES BulkAPIv1Insert run date: 2023-12-08 ------------------
20:30:00: Using Env|Schema: DEMO|dbo
20:30:00: Starting Loader for Account batchsize 10000
20:30:00: SSId added to Account_BulkAPIv1_Wait_Serial_Insert
20:30:02: Connection method BULK & BULK API
20:30:02: Bulk API method WAIT:SERIAL
20:30:02: Columns checked against Salesforce metadata
20:30:03: Starting load for Account_BulkAPIv1_Wait_Serial_Insert
20:30:16: JobId: 7508d00000TtMlOAAV
20:30:17: Excluded: AccountNumber_Info is not available on object Account
20:30:17: Excluded: Name_Info is not available on object Account
20:30:17: Load complete: Success:100 Failure:0
----------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

_Return helper table

The batch to which a given row has been allocated is logged in the helper table created in the run that is your load table + "_Return", using the above example it is:

Account_BulkAPIv1_Wait_Serial_Insert_Return

As with the Insert load table itself, the newly created Id (via the BulkAPIv1Insert operation) is also provided as well as the Error column, paired with the originally submitted SSId).

ss_BulkAPILog table

The Job Id is also preserved in the ss_BulkAPILog table, written on each submission. Batch detail is not maintained on WAIT methods.

Checking the load table

The success or failure errors for each row will be automatically written back to the Error column and as with the SOAP API Insert operation, with BulkAPIv1Insert SQL Sales is able to pass back the newly created Id, paired with the submitted SSId.

```
select * from Account_BulkAPIv1_Wait_Serial_Insert
```

sults | Messages

Id	Error	Name	Name_Info	AccountNumber_Info	SSId
0018d00000rSBuOAA...	Success	DEMO__PREFIXAAAAAAgggggggggg	AAAAAAgggggggggg	NULL	1
0018d00000rSBuPAAW	Success	DEMO__PREFIXNew Name, Upsert Demo 123	New Name, Upsert Demo 123	CD451796	2
0018d00000rSBuQAA...	Success	DEMO__PREFIXNew Name, Upsert Demo 789	New Name, Upsert Demo 789	CD656092	3
0018d00000rSBuRAA...	Success	DEMO__PREFIXAAAAAA46	AAAAAA46	CD439877	4
0018d00000rSBuSAAW	Success	DEMO__PREFIXNameChange7gAAA	NameChange7gAAA	CD355118	5
0018d00000rSBuTAAW	Success	DEMO__PREFIXNameChange7gAAA	NameChange7gAAA	CC947211	6
0018d00000rSBuUAA...	Success	DEMO__PREFIXNameChange7gAAA	NameChange7gAAA	CD736025	7
0018d00000rSBuVAAW	Success	DEMO__PREFIXMar 7 2023 3:20PM	Mar 7 2023 3:20PM	CD355119-A	8
0018d00000rSBuWA...	Success	DEMO__PREFIXMar 7 2023 3:20PM	Mar 7 2023 3:20PM	CD355120-B	9
0018d00000rSBuXAAW	Success	DEMO__PREFIXMar 7 2023 3:20PM	Mar 7 2023 3:20PM	CC978213	10

Create a load table Example (BACK)

(Refer to previous sections for details on SERIAL and PARALLEL options)

This example creates a load table called "Account_BulkAPIv1_BACK_Serial_Insert". The Id and Error columns are mandatory as with all Loader tables although with a BACK method, they will not be written back following the Insert.

BACK (specifies BACKGROUND running) means the SQL process will submit the load data in the batches specified by batchsize, however unlike the WAIT method, the ss_Loader run will end once that's done as the use case here is that no further action is required on the part of the User (i.e. submitting the data to the Salesforce Bulk API is sufficient enough for the job in hand). If you need to examine the success/failure of the submitted rows, there is a follow up script you can run to have the response data returned to SQL Server and your load table (instructions follow in this section).

```
drop table if exists Account_BulkAPIv1_BACK_Serial_Insert
select top 100
convert(nchar(18),null) as Id
,convert(nvarchar(255),null) as Error
,'DEMO__PREFIX' + Name as Name
-----
,Name as Name_Info
,AccountNumber as AccountNumber_Info
```

```
into Account_BulkAPIv1_BACK_Serial_Example_Insert
from Account
```

Running the example (Step 1)

```
exec ss_Loader 'BulkAPIv1Insert(50)', 'DEMO',
'Account_BulkAPIv1_BACK_Serial_Insert','BACK:SERIAL'
```

Note for the purposes of this test example, a batchsize of (50) has been defined, given the payload is 100 rows, this will force the creation of two batches, to illustrate the handling of multiple batches

```
SQL-SALES BulkAPIv1Insert(50) run date: 2023-12-09 --------------
07:47:44: Using Env|Schema: DEMO|dbo
07:47:44: Starting Loader for Account batchsize 50
07:47:44: SSId added to Account_BulkAPIv1_BACK_Serial_Insert
07:47:47: Connection method BULK & BULK API
07:47:47: Bulk API method BACK:SERIAL
07:47:47: Columns checked against Salesforce metadata
07:47:47: Starting load for Account_BulkAPIv1_BACK_Serial_Insert
07:48:01: JobId: 7508d00000TtQjVAAV
07:48:01: BatchId: 7518d00000dBwxRAAS CreatedDate: 2023-12-09 07:47:48
07:48:01: BatchId: 7518d00000dBwe6AAC CreatedDate: 2023-12-09 07:47:49
07:48:01: BulkAPIv1Insert BACKGROUND completed successfully
----------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

Note unlike the WAIT method, simply running BACK will not populate the _Result table nor write back Id or Error data to your load table, see the next set of instructions for how to do this, however the Batch Id(s) are included in the output dump for information purposes

ss_BulkAPILog table

The Job Id is also preserved in the ss_BulkAPILog table, written on each submission, for BACK methods, Batch information (Id and CreatedDate) is also written.

Running the example (Step 2 Option 1)

At any time after you have run the initial BACK, you can retrieve processed rows by reverting to using the WAIT method (either in SERIAL or PARALLEL mode). This is achieved by passing in the known Job Id into the @Special2 input parameter.

Note, when using “Option 1” if you have attempted to return processed rows “too soon” and some rows for a given Batch or Batches are not yet processed by Salesforce, SQL Sales will not be able to return an Id and Error value hence caution should be exercised and for you to check your load table.

Alternatively, you can use “Option 2” to check the status of your Job by submitting a followup BACK request, alongside your known Job Id. This will instruct SQL Sales to check all related Batches and return the status of the Job Id. When the Job is Closed, no further processing will occur by Salesforce and you can now run with WAIT to return all Id and Error values.

Note, running WAIT subsequently to the initial BACK for Update, Delete, Harddelete operations will return load status values to the Load table Error column. Whereas for Insert or Upsert, this is not possible, however for all operations, the _Return table is written back to, including new Ids and Error column values in the case of Insert or Upsert. The key difference is that the newly created or upserted to Ids are not tied back to the original load table row.

```
exec ss_Loader 'BulkAPIv1Insert', 'DEMO',
'Account_BulkAPIv1_BACK_Serial_Insert','JOB:WAIT:SERIAL','7508d00000TtQjVAAV'
```

```
SQL-SALES BulkAPIv1Insert run date: 2023-12-09 ------------------
08:05:12: Using Env|Schema: DEMO|dbo
08:05:12: Starting Loader for Account batchsize 10000
08:05:12: SSId added to Account_BulkAPIv1_BACK_Serial_Insert
08:05:15: Connection method BULK & BULK API
08:05:15: Bulk API method JOB:WAIT:SERIAL Job = 7508d00000TtQjVAAV
08:05:15: Columns checked against Salesforce metadata
08:05:15: Starting load for Account_BulkAPIv1_BACK_Serial_Insert
08:05:19: JobId: 7508d00000TtQjVAAV
08:05:19: Excluded: AccountNumber_Info is not available on object Account
08:05:19: Excluded: Name_Info is not available on object Account
08:05:19: Load complete: Success:100 Failure:0
-----------------------------------------------------------------
```

_Return helper table

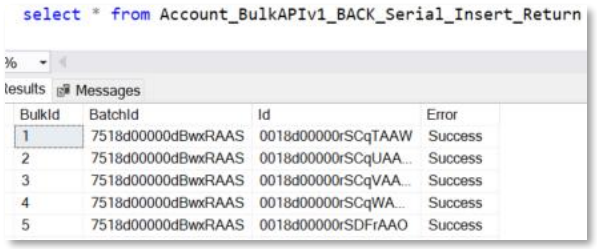
The batch to which a given Id has been allocated is logged in the helper table created in the run that is your load table + “_Return”, using the above example it is:

Account_BulkAPIv1_BACK_Serial_Insert_Return

As with the Insert load table itself, the newly created Id (via the BulkAPIv1Insert operation) is also provided as well as the Error column, although for Insert and Upsert operations, these rows do not correlate to your load table, you would have to run with the WAIT method if you require this. For

Update, Delete and Harddelete operations, the Result rows will correspond directly with your load table rows.

```
select * from Account_BulkAPIv1_BACK_Serial_Insert_Return
```



BulkId	BatchId	Id	Error
1	7518d00000dBwxRAAS	0018d00000rSCqTAAW	Success
2	7518d00000dBwxRAAS	0018d00000rSCqUAA...	Success
3	7518d00000dBwxRAAS	0018d00000rSCqVAA...	Success
4	7518d00000dBwxRAAS	0018d00000rSCqWA...	Success
5	7518d00000dBwxRAAS	0018d00000rSDFrAAO	Success

Checking the load table (Update, Delete, Harddelete)

The success or failure errors for each row will be automatically written back to the Error column and as with the SOAP API Insert operation, with BulkAPIv1Insert SQL Sales is able to pass back the newly created Id, paired with the submitted SSId.

Running the example (Step 2 Option 2)

You can keep submitting with BACK and the known Job Id until the Status shows that the Job has Closed (this will work with either SERIAL or PARALLEL). Once the Job is Closed you can run as with Option 1.

```
exec ss_Loader 'BulkAPIv1Insert(50)', 'DEMO',
'Account_BulkAPIv1_BACK_Serial_Insert','JOB:BACK:SERIAL','7508d00000TtQjVAAV'
```

```
SQL-SALES BulkAPIv1Insert(50) run date: 2023-12-09 --------------
08:41:26: Using Env|Schema: DEMO|dbo
08:41:26: Starting Loader for Account batchsize 50
08:41:26: SSId added to Account_BulkAPIv1_BACK_Serial_Insert
08:41:28: Connection method BULK & BULK API
08:41:28: Bulk API method JOB:BACK:SERIAL Job = 7508d00000TtQjVAAV
08:41:28: Columns checked against Salesforce metadata
08:41:29: Starting load for Account_BulkAPIv1_BACK_Serial_Insert
08:41:32: JobId: 7508d00000TtQjVAAV, Job Closed
08:41:32: BulkAPIv1Insert BACKGROUND completed successfully
----------------------------------------------------------------
```

BulkAPIv1Update

Create a load table Example (WAIT)

(Refer to previous sections for details on SERIAL and PARALLEL options as well as WAIT and BACK methods).

This example creates a load table called "Account_BulkAPIv1_Wait_Serial_Update". The Id and Error columns are mandatory. Generally, an update payload would be built up from querying or working

with prior replicated data for the affected Object by using ss_Replica / ss_Delta. All load operations require the Error column, whether success or failure, to guide and inform you.

```
drop table if exists Account_BulkAPIv1_Wait_Serial_Update
select top 100
convert(nchar(18),Id) as Id
,convert(nvarchar(255),null) as Error
,Id as AccountNumber
,AccountNumber as AccountNumber_Orig
into Account_BulkAPIv1_Wait_Serial_Update
from Account
where AccountNumber is null
order by createddate desc
```

Running the example

```
exec ss_Loader 'BulkAPIv1Update', 'DEMO', 'Account_BulkAPIv1_Wait_Serial_Update','WAIT:SERIAL'
```

```
SQL-SALES BulkAPIv1Update run date: 2023-12-09 ------------------
09:00:16: Using Env|Schema: DEMO|dbo
09:00:16: Starting Loader for Account batchsize 10000
09:00:16: SSId added to Account_BulkAPIv1_Wait_Serial_Update
09:00:19: Connection method BULK & BULK API
09:00:19: Bulk API method WAIT:SERIAL
09:00:19: Columns checked against Salesforce metadata
09:00:19: Starting load for Account_BulkAPIv1_Wait_Serial_Update
09:00:34: JobId: 7508d00000TtQsPAAV
09:00:34: Excluded: AccountNumber_Orig is not available on object Account
09:00:34: Load complete: Success:100 Failure:0
------------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

_Return helper table

The batch to which a given row has been allocated is logged in the helper table created in the run that is your load table + "_Return", using the above example it is:

Account_BulkAPIv1_Wait_Serial_Update_Return

As with the Update load table itself, the Error column is provided, paired with the originally submitted SSId.

```
select * from Account_BulkAPIv1_Wait_Serial_Update_Return
```

esults Messages

SSId	BatchId	Id	Error
1	7518d00000dBxbnA...	0018d00000rSDDDAA4	Success
2	7518d00000dBxbnA...	0018d00000rSDDEAA4	Success
3	7518d00000dBxbnA...	0018d00000rSDDFAA4	Success
4	7518d00000dBxbnA...	0018d00000rSDDGA...	Success
5	7518d00000dBxbnA...	0018d00000rSDDHAA4	Success

ss_BulkAPILog table

The Job Id is also preserved in the ss_BulkAPILog table, written on each submission.

Checking the load table

The success or failure errors for each row will be automatically written back to the Error column as with the SOAP API Update operation.

```
select * from Account_BulkAPIv1_Wait_Serial_Update
```

esults Messages

Id	Error	AccountNumber	AccountNumber_Orig	SSId
0018d00000rSDDDAA4	Success	0018d00000rSDDDAA4	NULL	1
0018d00000rSDDEAA4	Success	0018d00000rSDDEAA4	NULL	2
0018d00000rSDDFAA4	Success	0018d00000rSDDFAA4	NULL	3
0018d00000rSDDGAA4	Success	0018d00000rSDDGA...	NULL	4
0018d00000rSDDHAA4	Success	0018d00000rSDDHAA4	NULL	5

Create an Update load table Example (BACK)

(Refer to previous sections for details on SERIAL and PARALLEL options)

This example creates a load table called "Account_BulkAPIv1_BACK_Serial_Update". The Id and Error columns are mandatory. All load operations require the Error column, whether success or failure, to guide and inform you.

```
exec ss_Delta 'DEMO', 'Account'

drop table if exists Account_BulkAPIv1_BACK_Serial_Update
select top 100
convert(nchar(18),Id) as Id
,convert(nvarchar(255),null) as Error
,Id as AccountNumber
,AccountNumber as AccountNumber_Orig
into Account_BulkAPIv1_BACK_Serial_Update
from Account
where AccountNumber is null
order by createddate desc
```

Running the example (Step 1)

```
exec ss_Loader 'BulkAPIv1Update', 'DEMO', 'Account_BulkAPIv1_BACK_Serial_Update','BACK:SERIAL'
```

```
SQL-SALES BulkAPIv1Update run date: 2023-12-09 ------------------
09:09:32: Using Env|Schema: DEMO|dbo
09:09:32: Starting Loader for Account batchsize 10000
09:09:32: SSId added to Account_BulkAPIv1_BACK_Serial_Update
09:09:35: Connection method BULK & BULK API
09:09:35: Bulk API method BACK:SERIAL
09:09:35: Columns checked against Salesforce metadata
09:09:35: Starting load for Account_BulkAPIv1_BACK_Serial_Update
09:09:48: JobId: 7508d00000TtR82AAF
09:09:48: BatchId: 7518d00000dBxFdAAK CreatedDate: 2023-12-09 09:09:36
09:09:48: BulkAPIv1Update BACKGROUND completed successfully
----------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

Note unlike the WAIT method, simply running BACK will not populate the _Result table nor write back Error data to your load table, see the next set of instructions for how to do this, however the Batch Id(s) are included in the output dump for information purposes

ss_BulkAPILog table

The Job Id is also preserved in the ss_BulkAPILog table, written on each submission.

Running the example (Step 2 Option 1)

At any time after you have run the initial BACK, you can retrieve processed rows to your load table by reverting back to using the WAIT method (either in SERIAL or PARALLEL mode). This is achieved by passing in the known Job Id into the @Special2 input parameter.

Note, when using “Option 1” if you have attempted to return processed rows “too soon” and some rows for a given Batch or Batches are not yet processed by Salesforce, SQL Sales will not be able to return an Error value hence caution should be exercised and for you to check your load table.

Alternatively, you can use “Option 2” to check the status of your Job by submitting a followup BACK request, alongside your known Job Id. This will instruct SQL Sales to check all related Batches and return the status of the Job Id. When the Job is Closed, no further processing will occur by Salesforce and you can now run with WAIT to return all Id and Error values.

```
exec ss_Loader 'BulkAPIv1Update', 'DEMO',
'Account_BulkAPIv1_BACK_Serial_Update','JOB:WAIT:SERIAL','7508d00000TtR82AAF'
```

```
SQL-SALES BulkAPIv1Update run date: 2023-12-09 ------------------
09:13:41: Using Env|Schema: DEMO|dbo
09:13:41: Starting Loader for Account batchsize 10000
09:13:41: SSId added to Account_BulkAPIv1_BACK_Serial_Update
09:13:44: Connection method BULK & BULK API
09:13:44: Bulk API method JOB:WAIT:SERIAL Job = 7508d00000TtR82AAF
09:13:44: Columns checked against Salesforce metadata
09:13:44: Starting load for Account_BulkAPIv1_BACK_Serial_Update
09:13:48: JobId: 7508d00000TtR82AAF, Job Closed
09:13:49: Excluded: AccountNumber_Orig is not available on object Account
09:13:49: Load complete: Success:100 Failure:0
-----------------------------------------------------------------
```

_Return helper table

The batch to which a given row has been allocated is logged in the helper table created in the run that is your load table + “_Batch”, using the above example it is:

Account_BulkAPIv1_BACK_Serial_Update_Return

As with the Update load table itself, the Error column is provided, paired with the originally submitted SSId.

Checking the load table

The success or failure errors for each row will be automatically written back to the Error column as with the SOAP API Update operation.

Running the example (Step 2 Option 2)

You can keep submitting with BACK and the known Job Id until the Status shows that the Job has Closed (this will work with either SERIAL or PARALLEL). Once the Job is Closed you can run as with Option 1.

BulkAPIv1Delete

Create a load table Example (WAIT)

(Refer to previous sections for details on SERIAL and PARALLEL options as well as WAIT and BACK methods).

This example creates a load table called "Account_BulkAPIv1_Wait_Serial_Delete". The Id and Error columns are mandatory. Generally, a delete payload would be built up from querying or working with prior replicated data for the affected Object by using ss_Replica / ss_Delta. All load operations require the Error column, whether success or failure, to guide and inform you.

```
exec ss_Delta 'DEMO', 'Account'

drop table if exists Account_BulkAPIv1_Wait_Serial_Delete
select top 100
convert(nchar(18),Id) as Id
,convert(nvarchar(255),null) as Error
,Name as Name_Info
into Account_BulkAPIv1_Wait_Serial_Delete
from Account
order by createddate desc
```

Running the example

```
exec ss_Loader 'BulkAPIv1Delete', 'DEMO', 'Account_BulkAPIv1_Wait_Serial_Delete','WAIT:SERIAL'
```

```
SQL-SALES BulkAPIv1Delete run date: 2023-12-09 ------------------
09:25:17: Using Env|Schema: DEMO|dbo
09:25:18: Starting Loader for Account batchsize 10000
09:25:18: SSId added to Account_BulkAPIv1_Wait_Serial_Delete
09:25:20: Connection method BULK & BULK API
09:25:20: Bulk API method WAIT:SERIAL
09:25:20: Columns checked against Salesforce metadata
09:25:21: Starting load for Account_BulkAPIv1_Wait_Serial_Delete
09:25:35: JobId: 7508d00000TtRBUAA3
09:25:35: Load complete: Success:100 Failure:0
-----------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

_Return helper table

The batch to which a given row has been allocated is logged in the helper table created in the run that is your load table + "_Batch", using the above example it is:

Account_BulkAPIv1_Wait_Serial_Delete_Return

As with the Delete load table itself, the Error column is provided, paired with the originally submitted SSId.

```
select * from Account_BulkAPIv1_Wait_Serial_Delete_Return
```

sults | Messages

SSId	BatchId	Id	Error
1	7518d00000dBxOpA...	0018d00000rSDPwAAO	Success
2	7518d00000dBxOpA...	0018d00000rSDPxAAO	Success
3	7518d00000dBxOpA...	0018d00000rSDPyAAO	Success
4	7518d00000dBxOpA...	0018d00000rSDPzAAO	Success
5	7518d00000dBxOpA...	0018d00000rSDQ0AAO	Success

ss_BulkAPILog table

The Job Id is also preserved in the ss_BulkAPILog table, written on each submission.

Checking the load table

The success or failure errors for each row will be automatically written back to the Error column as with the SOAP API Delete operation.

```
select * from Account_BulkAPIv1_Wait_Serial_Delete
```

sults | Messages

Id	Error	Name_Info	SSId
0018d00000rSDPwAAO	Success	DEMO__PREFIX41	1
0018d00000rSDPxAAO	Success	DEMO__PREFIX42	2
0018d00000rSDPyAAO	Success	DEMO__PREFIX68	3
0018d00000rSDPzAAO	Success	DEMO__PREFIX43	4
0018d00000rSDQ0AAO	Success	DEMO__PREFIX69	5

Create a load table Example (BACK)

(Refer to previous sections for details on SERIAL and PARALLEL options)

This example creates a load table called "Account_BulkAPIv1_BACK_Serial_Delete". The Id and Error columns are mandatory. All load operations require the Error column, whether success or failure, to guide and inform you.

```
exec ss_Delta 'DEMO', 'Account'

drop table if exists Account_BulkAPIv1_BACK_Serial_Delete
select top 100
convert(nchar(18),Id) as Id
,convert(nvarchar(255),null) as Error
,Name as Name_Info
into Account_BulkAPIv1_BACK_Serial_Delete
from Account
order by createddate desc
```

Running the example (Step 1)

```
exec ss_Loader 'BulkAPIv1Delete', 'DEMO', 'Account_BulkAPIv1_BACK_Serial_Delete','BACK:SERIAL'
```

```
SQL-SALES BulkAPIv1Delete run date: 2023-12-09 ------------------
19:35:22: Using Env|Schema: DEMO|dbo
19:35:22: Starting Loader for Account batchsize 10000
19:35:22: SSId added to Account_BulkAPIv1_BACK_Serial_Delete
19:35:25: Connection method BULK & BULK API
19:35:25: Bulk API method BACK:SERIAL
19:35:25: Columns checked against Salesforce metadata
19:35:26: Starting load for Account_BulkAPIv1_BACK_Serial_Delete
19:35:39: JobId: 7508d00000TtTgVAAV
19:35:39: BatchId: 7518d00000dC1LzAAK CreatedDate: 2023-12-09 19:35:29
19:35:39: BulkAPIv1Delete BACKGROUND completed successfully
----------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

Note unlike the WAIT method, simply running BACK will not populate the _Result table nor write back Error data to your load table, see the next set of instructions for how to do this, however the Batch Id(s) are included in the output dump for information purposes

ss_BulkAPILog table

The Job Id is also preserved in the Ss_BulkAPILog table, written on each submission.

Running the example (Step 2 Option 1)

At any time after you have run the initial BACK, you can retrieve processed rows to your load table by reverting back to using the WAIT method (either in SERIAL or PARALLEL mode). This is achieved by passing in the known Job Id into the @Special2 input parameter.

Note, when using "Option 1" if you have attempted to return processed rows "too soon" and some rows for a given Batch or Batches are not yet processed by Salesforce, SQL Sales will not be able to return an Error value hence caution should be exercised and for you to check your load table.

Alternatively, you can use "Option 2" to check the status of your Job by submitting a followup BACK request, alongside your known Job Id. This will instruct SQL Sales to check all related Batches and return the status of the Job Id. When the Job is Closed, no further processing will occur by Salesforce and you can now run with WAIT to return all Id and Error values.

```
exec ss_Loader 'BulkAPIv1Delete', 'DEMO',
'Account_BulkAPIv1_BACK_Serial_Delete','JOB:WAIT:SERIAL','7508d00000TtTgVAAV'
```

```
SQL-SALES BulkAPIv1Delete run date: 2023-12-09 ------------------
19:41:47: Using Env|Schema: DEMO|dbo
19:41:47: Starting Loader for Account batchsize 10000
19:41:48: SSId added to Account_BulkAPIv1_BACK_Serial_Delete
19:41:50: Connection method BULK & BULK API
19:41:50: Bulk API method JOB:WAIT:SERIAL Job = 7508d00000TtTgVAAV
19:41:50: Columns checked against Salesforce metadata
19:41:51: Starting load for Account_BulkAPIv1_BACK_Serial_Delete
19:41:54: JobId: 7508d00000TtTgVAAV, Job Closed
19:41:54: Load complete: Success:100 Failure:0
---------------------------------------------------------------
```

_Return helper table

The batch to which a given row has been allocated is logged in the helper table created in the run that is your load table + "_Batch", using the above example it is:

Account_BulkAPIv1_BACK_Serial_Delete_Return

As with the Delete load table itself, the Error column is provided, paired with the originally submitted SSId.

```
select * from Account_BulkAPIv1_FireForget_Serial_Example_Delete_Batch
```

sults Messages

SSId	BatchId	Id	Error
1	7518d00000bkNYEAA2	0018d00000rQHOaAAO	Success
2	7518d00000bkNYEAA2	0018d00000rQHObAAO	Success
3	7518d00000bkNYEAA2	0018d00000rQHOcAAO	Success
4	7518d00000bkNYEAA2	0018d00000rQHOdAAO	Success
5	7518d00000bkNYEAA2	0018d00000rQHOeAAO	Success

Checking the load table

The success or failure errors for each row will be automatically written back to the Error column as with the SOAP API Delete operation.

Running the example (Step 2 Option 2)

You can keep submitting with BACK and the known Job Id until the Status shows that the Job has Closed (this will work with either SERIAL or PARALLEL). Once the Job is Closed you can run as with Option 1.

BulkAPIv1Harddelete

Create a load table Example (WAIT)

(Refer to previous sections for details on SERIAL and PARALLEL options as well as WAIT and BACK methods).

```
exec ss_Delta 'DEMO', 'Account'

drop table if exists Account_BulkAPIv1_Wait_Serial_Harddelete
select top 100
convert(nchar(18),Id) as Id
,convert(nvarchar(255),null) as Error
,Name as Name_Info
into Account_BulkAPIv1_Wait_Serial_Harddelete
from Account
order by createddate desc
```

Running the example

```
exec ss_Loader 'BulkAPIv1Harddelete', 'DEMO',
'Account_BulkAPIv1_Wait_Serial_Harddelete','WAIT:SERIAL'
```

```
SQL-SALES BulkAPIv1Harddelete run date: 2023-12-09 --------------
20:00:23: Using Env|Schema: DEMO|dbo
20:00:23: Starting Loader for Account batchsize 10000
20:00:23: SSId added to Account_BulkAPIv1_Wait_Serial_Harddelete
20:00:26: Connection method BULK & BULK API
20:00:26: Bulk API method WAIT:SERIAL
20:00:26: Columns checked against Salesforce metadata
20:00:26: Starting load for Account_BulkAPIv1_Wait_Serial_Harddelete
20:00:28: Failed to create job. Response: {'exceptionCode': 'FeatureNotEnabled',
'exceptionMessage': 'hardDelete operation requires special user profile permission, please
contact your system administrator'}
----------------------------------------------------------------
```

Note, by default a sys admin user will not have permission to run Harddelete, you will need a special profile created (in fact cloned) from a system administrator and Hard delete enabled on it, without this initial preparation you will encounter the above error message

```
exec ss_Loader 'BulkAPIv1Harddelete', 'HARD',
'Account_BulkAPIv1_Wait_Serial_Harddelete','WAIT:SERIAL'
```

A special Environment has been setup in this Demo, called "HARD" (which uses a different username, with a Harddelete profile enabled

```
SQL-SALES BulkAPIv1Harddelete run date: 2023-12-09 --------------
20:09:50: Using Env|Schema: HARD|dbo
20:09:50: Starting Loader for Account batchsize 10000
20:09:50: SSId added to Account_BulkAPIv1_Wait_Serial_Harddelete
20:09:53: Connection method BULK & BULK API
20:09:53: Bulk API method WAIT:SERIAL
20:09:53: Columns checked against Salesforce metadata
20:09:53: Starting load for Account_BulkAPIv1_Wait_Serial_Harddelete
20:10:07: JobId: 7508d00000TtTqeAAF
20:10:08: Load complete: Success:100 Failure:0
-----------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

_Return helper table

The batch to which a given row has been allocated is logged in the helper table created in the run that is your load table + "_Batch", using the above example it is:

Account_BulkAPIv1_Wait_Serial_Harddelete_Return

The Error column is provided, paired with the originally submitted SSId.

```
select * from Account_BulkAPIv1_Wait_Serial_Harddelete_Return
```

sults | Messages

SSId	BatchId	Id	Error
1	7518d00000dC1bDAAS	0018d00000rSDKcAAO	Success
2	7518d00000dC1bDAAS	0018d00000rSDKdAAO	Success
3	7518d00000dC1bDAAS	0018d00000rSDKeAAO	Success
4	7518d00000dC1bDAAS	0018d00000rSDKfAAO	Success
5	7518d00000dC1bDAAS	0018d00000rSDKgAAO	Success

Ss_BulkAPILog table

The Job Id is also preserved in the Ss_BulkAPILog table, written on each submission.

Checking the load table

The success or failure errors for each row will be automatically written back to the Error column.

```
select * from Account_BulkAPIv1_Wait_Serial_Harddelete
```

sults | Messages

Id	Error	Name_Info	SSId
0018d00000rSDKcAAO	Success	DEMO__PREFIXAAAAAAgggggggggg	1
0018d00000rSDKdAAO	Success	DEMO__PREFIXNew Name, Upsert Demo 123	2
0018d00000rSDKeAAO	Success	DEMO__PREFIXNew Name, Upsert Demo 789	3
0018d00000rSDKfAAO	Success	DEMO__PREFIXAAAAAA46	4
0018d00000rSDKgAAO	Success	DEMO__PREFIXNameChange7gAAA	5

Create a load table Example (BACK)

(Refer to previous sections for details on SERIAL and PARALLEL options)

This example creates a load table called "Account_BulkAPIv1_BACK_Serial_Harddelete". The Id and Error columns are mandatory. All load operations require the Error column, whether success or failure, to guide and inform you.

```
exec ss_Delta 'DEMO', 'Account'

drop table if exists Account_BulkAPIv1_BACK_Serial_Harddelete
select top 100
convert(nchar(18),Id) as Id
,convert(nvarchar(255),null) as Error
,Name as Name_Info
into Account_BulkAPIv1_BACK_Serial_Harddelete
from Account
order by createddate desc
```

Running the example (Step 1)

```
exec ss_Loader 'BulkAPIv1Harddelete', 'HARD',
'Account_BulkAPIv1_BACK_Serial_Harddelete','BACK:SERIAL'
```

```
SQL-SALES BulkAPIv1Harddelete run date: 2023-12-09 --------------
20:21:44: Using Env|Schema: HARD|dbo
20:21:44: Starting Loader for Account batchsize 10000
20:21:44: SSId added to Account_BulkAPIv1_BACK_Serial_Harddelete
20:21:47: Connection method BULK & BULK API
20:21:47: Bulk API method BACK:SERIAL
20:21:47: Columns checked against Salesforce metadata
20:21:47: Starting load for Account_BulkAPIv1_BACK_Serial_Harddelete
20:21:59: JobId: 7508d00000TtTt4AAF
20:21:59: BatchId: 7518d00000dC1eHAAS CreatedDate: 2023-12-09 20:21:49
20:21:59: BulkAPIv1Harddelete BACKGROUND completed successfully
----------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

Note unlike the WAIT method, simply running BACK will not populate the _Return table nor write back Error data to your load table, see the next set of instructions for how to do this, however the Batch Id(s) are included in the output dump for information purposes

Ss_BulkAPILog table

The Job Id is also preserved in the Ss_BulkAPILog table, written on each submission.

Running the example (Step 2 Option 1)

At any time after you have run the initial BACK, you can retrieve processed rows to your load table by reverting back to using the WAIT method (either in SERIAL or PARALLEL mode). This is achieved by passing in the known Job Id into the @Special2 input parameter.

Note, when using “Option 1” if you have attempted to return processed rows “too soon” and some rows for a given Batch or Batches are not yet processed by Salesforce, SQL Sales will not be able to return an Error value hence caution should be exercised and for you to check your load table.

Alternatively, you can use “Option 2” to check the status of your Job by submitting a followup BACK request, alongside your known Job Id. This will instruct SQL Sales to check all related Batches and return the status of the Job Id. When the Job is Closed, no further processing will occur by Salesforce and you can now run with WAIT to return all Id and Error values.

```
exec ss_Loader 'BulkAPIv1Harddelete', 'HARD',
'Account_BulkAPIv1_BACK_Serial_Harddelete','JOB:WAIT:SERIAL','7508d00000TtTt4AAF'
```

```
SQL-SALES BulkAPIv1Harddelete run date: 2023-12-09 --------------
20:26:05: Using Env|Schema: HARD|dbo
20:26:05: Starting Loader for Account batchsize 10000
20:26:05: SSId added to Account_BulkAPIv1_BACK_Serial_Harddelete
20:26:08: Connection method BULK & BULK API
20:26:08: Bulk API method JOB:WAIT:SERIAL Job = 7508d00000TtTt4AAF
20:26:08: Columns checked against Salesforce metadata
20:26:08: Starting load for Account_BulkAPIv1_BACK_Serial_Harddelete
20:26:12: JobId: 7508d00000TtTt4AAF, Job Closed
20:26:12: Load complete: Success:100 Failure:0
-----------------------------------------------------------------
```

_Return helper table

The batch to which a given row has been allocated is logged in the helper table created in the run that is your load table + "_Return", using the above example it is:

Account_BulkAPIv1_BACK_Serial_Harddelete_Return

As with the Update and Delete operation load table itself, the Error column is provided, paired with the originally submitted SSId.

```
select * from Account_BulkAPIv1_BACK_Serial_Harddelete_Return
```

sults | Messages

Bulk...	BatchId	Id	Error
1	7518d00000dC1eHAAS	0018d00000rSCqTAAW	Success
2	7518d00000dC1eHAAS	0018d00000rSCqUAA...	Success
3	7518d00000dC1eHAAS	0018d00000rSCqVAA...	Success
4	7518d00000dC1eHAAS	0018d00000rSCqWA...	Success
5	7518d00000dC1eHAAS	0018d00000rSD6bAAG	Success

Checking the load table

The success or failure errors for each row will be automatically written back to the Error column.

```
select * from Account_BulkAPIv1_BACK_Serial_Harddelete
```

sults | Messages

Id	Error	Name_Info	SSId
0018d00000rSCqTAAW	Success	DEMO__PREFIXAAAAAAgggggggggg	1
0018d00000rSCqUAA...	Success	DEMO__PREFIXNew Name, Upsert Demo 1...	2
0018d00000rSCqVAAW	Success	DEMO__PREFIXNew Name, Upsert Demo 7...	3
0018d00000rSCqWA...	Success	DEMO__PREFIXAAAAAA46	4
0018d00000rSD6bAAG	Success	DEMO__PREFIX41	5

Running the example (Step 2 Option 2)

You can keep submitting with BACK and the known Job Id until the Status shows that the Job has Closed (this will work with either SERIAL or PARALLEL). Once the Job is Closed you can run as with Option 1.

BulkAPIv1Upsert

As with the SOAP API Upsert, working with the Salesfore Bulk API v1 is a little different to the other operations. BulkAPIv1Update, BulkAPIv1Delete and BulkAPIv1Harddelete work from the provided Id, with regards what records to operate against. Insert merely creates new records and passes the new Id back.

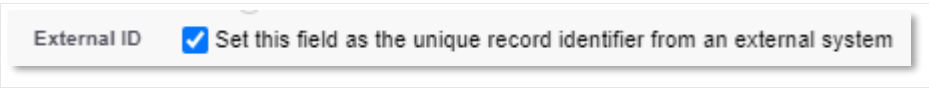
Whereas BulkAPIv1Upsert, as the name suggests, works primarily off a provided External Id for the given object being upserted against.

If a match is found in that Salesforce object for the value being passed in, then the Operation will update the provided fields in the table payload to the matched record.

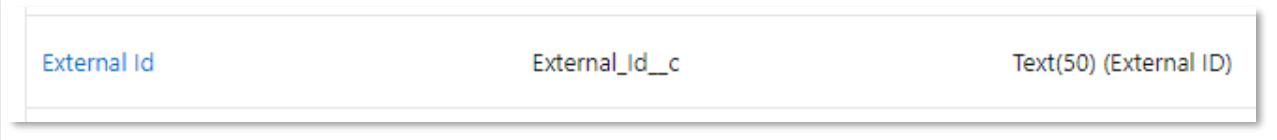
If a match is not found, the Upsert operation will insert a new record, using the provided fields in the table payload. All load operations require the Error column, whether success or failure, to guide and inform you.

External Id

Upsert will only work against an External data type field. This is a special setting for a field in Salesforce:



For this example, the field "External_Id__c" has been added to the Account object in our Demo Org, note the special indicator "(External ID)". If your intended External Id field does not have this, it is likely not actually setup as an External Id, no matter what the field name is.



SQL Sales will inform if it is not truly an External Id, this is demonstrated in the following examples.

Create a load table Example (WAIT)

(Refer to previous sections for details on SERIAL and PARALLEL options as well as WAIT and BACK methods).

This example creates a load table called "Account_BulkAPIv1_Wait_Serial_Upsert".

```
drop table if exists Account_BulkAPIv1_Wait_Serial_Upsert
select top 33
convert(nchar(18),null) as Id
,convert(nvarchar(255),null) as Error
,Name
,'UPDATE_TEST_' + Id as AccountNumber
,External_Id__c
,AccountNumber as AccountNumber_Orig
into Account_BulkAPIv1_Wait_Serial_Upsert
from Account
where AccountNumber not like '0%'
and External_Id__c is not null
order by createddate desc

insert Account_BulkAPIv1_Wait_Serial_Upsert
(Id
,Error
,Name
,AccountNumber
,External_Id__c
,AccountNumber_Orig)

select top 33
convert(nchar(18),null) --Id
,convert(nvarchar(255),null) --Error
,Name
,'REJECT_TEST_' + Id --AccountNumber
,null --External_Id__c
,null --AccountNumber_Orig
from Account
where AccountNumber not like '0%'
and External_Id__c is not null
order by createddate desc

insert Account_BulkAPIv1_Wait_Serial_Upsert
(Id
,Error
,Name
,AccountNumber
,External_Id__c
,AccountNumber_Orig)

select top 33
convert(nchar(18),null) --Id
,convert(nvarchar(255),null) --Error
,Name
,'INSERT_TEST_' + Id --AccountNumber
,'INSERT_TEST_' + Id --AExternal_Id__c
,null --AccountNumber_Orig
from Account
where AccountNumber not like '0%'
and External_Id__c is not null
order by createddate desc
```

Check the Payload

Note, those records with a matched (to Salesforce) value in External_Id__c will result in an update, those with a new value but no match to Salesforce will result in an Insert, whereas those with no value in External_Id__c will be rejected (as External_Id__c, being the specified External Id field for the BulkAPIv1Upsert operation, es expected to be populated).

```
select * from Account_BulkAPIv1_Wait_Serial_Upsert
order by name
```

sults Messages

Id	Error	Name	AccountNumber	External_Id__c	AccountNumber_Orig
NULL	NU...	20230920av1__0018d00000P4XctAAF	UPDATE_TEST_0018d00000rSEpqAAG	Test0018d00000P4XctAAF	20230921av1__0018d00000P4XctAAF
NULL	NU...	20230920av1__0018d00000P4XctAAF	INSERT_TEST_0018d00000rSEpqAAG	INSERT_TEST_0018d00000rSEpqA...	NULL
NULL	NU...	20230920av1__0018d00000P4XctAAF	REJECT_TEST_0018d00000rSEpqAAG	NULL	NULL
NULL	NU...	20230920av1__0018d00000P8DFzAAN	REJECT_TEST_0018d00000rSEprAAG	NULL	NULL
NULL	NU...	20230920av1__0018d00000P8DFzAAN	INSERT_TEST_0018d00000rSEprAAG	INSERT_TEST_0018d00000rSEprAAG	NULL
NULL	NU...	20230920av1__0018d00000P8DFzAAN	UPDATE_TEST_0018d00000rSEprAAG	Test0018d00000P8DFzAAN	20230921av1__0018d00000P8DFzAAN
NULL	NU...	20230920av1__0018d00000P8DG0AAN	UPDATE_TEST_0018d00000rSEpsAAG	Test0018d00000P8DG0AAN	20230921av1__0018d00000P8DG0AAN
NULL	NU...	20230920av1__0018d00000P8DG0AAN	INSERT_TEST_0018d00000rSEpsAAG	INSERT_TEST_0018d00000rSEpsAA...	NULL
NULL	NU...	20230920av1__0018d00000P8DG0AAN	REJECT_TEST_0018d00000rSEpsAAG	NULL	NULL
NULL	NU...	20230920av1__0018d00000P8DG3AAN	REJECT_TEST_0018d00000rSEptAAG	NULL	NULL

Running the example (incorrect)

```
exec ss_Loader 'BulkAPIv1Upsert', 'DEMO', 'Account_BulkAPIv1_Wait_Serial_Upsert','WAIT:SERIAL'
```

```
SQL-SALES BulkAPIv1Upsert run date: 2023-12-09 ------------------
20:47:17: Using Env|Schema: DEMO|dbo
20:47:17: Provided Upsert Operation value not in the correct format, for a hypothetical field
called "External_Id__c"
20:47:17: set (in Salesforce) as an External Id (this is a field setting)
20:47:17: the expected input value for the standard web services API is:
Upsert:XId=External_Id__c
20:47:17: or BulkAPIv1Upsert:XId=External_Id__c for the bulk API v1
20:47:17: or BulkAPIv2Upsert:XId=External_Id__c for the bulk API v2
20:47:17: with a defined batchsize (for example 100 or 1000 respectively) these would be:
20:47:17: Upsert(100):XId=External_Id__c | BulkAPIv1Upsert(10000):XId=External_Id__c |
BulkAPIv2Upsert(2000):XId=External_Id__c
20:47:17: Note, the provided External Id is validated directly against Salesforce Account
20:47:17: prior to the run commencing to check it is actually defined as an External Id.
-----------------------------------------------------------------
```

Note, the format provided in the example is incorrect, observe the supporting help text

Running the example (correct)

```
exec ss_Loader 'BulkAPIv1Upsert:XId=External_Id__c', 'DEMO',
'Account_BulkAPIv1_Wait_Serial_Upsert','WAIT:SERIAL'
```

```
SQL-SALES BulkAPIv1Upsert:XId=External_Id__c run date: 2023-12-09
20:47:51: Using Env|Schema: DEMO|dbo
20:47:54: Starting Loader for Account batchsize 10000
20:47:54: SSId added to Account_BulkAPIv1_Wait_Serial_Upsert
20:47:57: Connection method BULK & BULK API
20:47:57: Bulk API method WAIT:SERIAL
20:47:57: Columns checked against Salesforce metadata
20:47:57: Starting load for Account_BulkAPIv1_Wait_Serial_Upsert
20:48:11: JobId: 7508d00000TtTzbAAF
20:48:12: Excluded: AccountNumber_Orig is not available on object Account
20:48:12: Load complete: Success:66 Failure:33
------------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

_Return helper table

The batch to which a given row has been allocated is logged in the helper table created in the run that is your load table + "_Return", using the above example it is:

Account_BulkAPIv1_Wait_Serial_Upsert_Return

For the load table itself as this was run with WAIT, the Error column is provided, paired with the originally submitted SSId.

```
select * from Account_BulkAPIv1_Wait_Serial_Upsert_Return
```

Results | Messages

SSId	BatchId	Id	Error
1	7518d00000dC1nnAAC	0018d00000rSEqcAAG	Success
2	7518d00000dC1nnAAC	0018d00000rSEqBAA...	Success
3	7518d00000dC1nnAAC	0018d00000rSEqbAAG	Success
4	7518d00000dC1nnAAC	0018d00000rSEqAAA...	Success
5	7518d00000dC1nnAAC	0018d00000rSEqaAAG	Success

Ss_BulkAPILog table

The Job Id is also preserved in the Ss_BulkAPILog table, written on each submission.

Checking the load table

The success or failure errors for each row will be automatically written back to the Error column as with the SOAP API Update operation.

```
select * from Account_BulkAPIv1_Wait_Serial_Upsert
```

sults Messages

Id	Error	Name	AccountNumber	External_Id__c	AccountNumber_Orig	SSId
0018d00000rSEpqAAG	Success	20230920av1__0018d00000P4XctAAF	UPDATE_TEST_0018d00000rSEpqAAG	Test0018d00000P4XctAAF	20230921av1__0018d00000P4XctAAF	25
0018d00000rSEsbAAG	Success	20230920av1__0018d00000P4XctAAF	INSERT_TEST_0018d00000rSEpqAAG	INSERT_TEST_0018d00000rSEpqAAG	NULL	50
	DUPLICATE_VALUE:Duplicate external i...	20230920av1__0018d00000P4XctAAF	REJECT_TEST_0018d00000rSEpqAAG	NULL	NULL	88
	DUPLICATE_VALUE:Duplicate external i...	20230920av1__0018d00000P8DFzAAN	REJECT_TEST_0018d00000rSEprAAG	NULL	NULL	87
0018d00000rSEsaAAG	Success	20230920av1__0018d00000P8DFzAAN	INSERT_TEST_0018d00000rSEprAAG	INSERT_TEST_0018d00000rSEprAAG	NULL	49

Further uses

Refer to the worked examples shown for the other BulkAPIv1 operations, they would equally apply to BulkAPIv1Upsert, ensure you take note of the guidance regarding the use of the XId switch.

BulkAPIv2Insert

Create a load table Example (WAIT)

This example creates a load table called "Account_BulkAPIv1_Wait_Example_Insert".

WAIT in Version 2 of the Bulk API means the SQL process will remain live while it waits for the Bulk API to return all responses to the _Return table (Version 2 is not able to return Error values and Ids back to the load table for Inserts and Updates, however it will for updates, Deletes and Harddeletes.

```
drop table if exists Account_BulkAPIv2_Wait_Insert
select top 100
convert(nchar(18),null) as Id
,convert(nvarchar(255),null) as Error
,'DEMO__PREFIXv2' + Name as Name
,'DEMO__PREFIXv2' + AccountNumber as AccountNumber
-----
,Name as Name_Info
,AccountNumber as AccountNumber_Info
into Account_BulkAPIv2_Wait_Insert
from Account
where AccountNumber is not null
```

Running the example

```
exec ss_Loader 'BulkAPIv2Insert', 'DEMO', 'Account_BulkAPIv2_Wait_Insert','WAIT'
```

```
SQL-SALES BulkAPIv2Insert run date: 2023-12-09 ------------------
21:18:44: Using Env|Schema: DEMO|dbo
21:18:44: Starting Loader for Account batchsize 10000
21:18:44: SSId added to Account_BulkAPIv2_Wait_Insert
21:18:47: Connection method BULK & BULK API
21:18:47: Bulk API method WAIT
21:18:47: Columns checked against Salesforce metadata
21:18:48: Starting load for Account_BulkAPIv2_Wait_Insert
21:19:03: JobId: 7508d00000TtU7kAAF
21:19:03: Excluded: AccountNumber_Info is not available on object Account
21:19:03: Excluded: Name_Info is not available on object Account
21:19:03: Load complete: Success:100 Failure:0
----------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

Ss_BulkAPILog table

The Job Id is also preserved in the Ss_BulkAPILog table, written on each submission.

Checking the load table

For Inserts and Version 2, no direct response of Ids and Error column values are written back to the Load table.

_Return helper table

The batch to which a given row has been allocated is logged in the helper table created in the run that is your load table + "_Return", using the above example it is:

Account_BulkAPIv2_Wait_Insert_Return

For Version 2, this provides the created Id and/or Error reason for information purposes, but the rows do not tire back to your load table.

Create a load table Example (BACK)

This example creates a load table called "Account_BulkAPIv2_BACK_Insert".

```
drop table if exists Account_BulkAPIv2_BACK_Insert
select top 100
convert(nchar(18),null) as Id
,convert(nvarchar(255),null) as Error
,'DEMO__PREFIX' + Name as Name
,'DEMO__PREFIX' + AccountNumber as AccountNumber
-----
,Name as Name_Info
,AccountNumber as AccountNumber_Info
into Account_BulkAPIv2_BACK_Insert
from Account
where AccountNumber is not null
```

Running the example (Step 1)

```
exec ss_Loader 'BulkAPIv2Insert', 'DEMO', 'Account_BulkAPIv2_BACK_Insert','BACK'
```

```
SQL-SALES BulkAPIv2Insert run date: 2023-12-09 ------------------
21:25:53: Using Env|Schema: DEMO|dbo
21:25:53: Starting Loader for Account batchsize 10000
21:25:54: SSId added to Account_BulkAPIv2_BACK_Insert
21:25:57: Connection method BULK & BULK API
21:25:57: Bulk API method BACK
21:25:57: Columns checked against Salesforce metadata
21:25:57: Starting load for Account_BulkAPIv2_BACK_Insert
21:26:00: JobId: 7508d00000TtU9WAAV
21:26:00: BulkAPIv2Insert BACKGROUND completed successfully
---------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

Note unlike the WAIT method, simply running BACK will not write back to the Result table, you will need to run a subsequent WAIT for that to be actioned

Ss_BulkAPILog table

The Job Id is also preserved in the Ss_BulkAPILog table, written on each submission.

Running the example (Step 2 Option 1)

At any time after you have run the initial BACK, you can retrieve processed rows to your _Return table by reverting back to using the WAIT method. This is achieved by passing in the known Job Id into the @Special2 input parameter.

Note, when using “Option 1” if you have attempted to return processed rows “too soon” and some rows for a given Batch or Batches are not yet processed by Salesforce, SQL Sales will not be able to return an Id and Error value hence caution should be exercised and for you to check your load table.

Alternatively, you can use “Option 2” to check the status of your Job by submitting a followup BULK request, alongside your known Job Id. This will instruct SQL Sales to check all related Batches and return the status of the Job Id. When the Job is Closed, no further processing will occur by Salesforce and you can now run with WAIT to return all Id and Error values.

```
exec ss_Loader 'BulkAPIv2Insert', 'DEMO',
'Account_BulkAPIv2_BACK_Insert','JOB:WAIT','7508d00000SopjWAAR'
```

```
SQL-SALES BulkAPIv2Insert run date: 2023-12-09 ------------------
21:29:59: Using Env|Schema: DEMO|dbo
21:29:59: Starting Loader for Account batchsize 10000
21:29:59: SSId added to Account_BulkAPIv2_BACK_Insert
21:30:02: Connection method BULK & BULK API
21:30:02: Bulk API method JOB:WAIT Job = 7508d00000TtU9WAAV
21:30:02: Columns checked against Salesforce metadata
21:30:03: Starting load for Account_BulkAPIv2_BACK_Insert
21:30:06: JobId: 7508d00000TtU9WAAV, Job Complete
21:30:06: Excluded: AccountNumber_Info is not available on object Account
21:30:06: Excluded: Name_Info is not available on object Account
21:30:06: Load complete: Success:100 Failure:0
-----------------------------------------------------------------
```

Checking the _Return table

The success or failure errors and newly created Ids will be automatically written back to the _Return table.

Running the example (Step 2 Option 2)

You can keep submitting with BACK and the known Job Id until the Status shows that the Job has Closed. Once the Job is Closed you can run as with Option 1.

```
exec ss_Loader 'BulkAPIv2Insert', 'DEMO', 'Account_BulkAPIv2_BACK_Insert','JOB:BACK','
7508d00000TtU9WAAV'
```

```
SQL-SALES BulkAPIv2Insert run date: 2023-12-09 ------------------
21:32:33: Using Env|Schema: DEMO|dbo
21:32:33: Starting Loader for Account batchsize 10000
21:32:33: SSId added to Account_BulkAPIv2_BACK_Insert
21:32:36: Connection method BULK & BULK API
21:32:36: Bulk API method JOB:BACK Job = 7508d00000TtU9WAAV
21:32:36: Columns checked against Salesforce metadata
21:32:36: Starting load for Account_BulkAPIv2_BACK_Insert
21:32:40: JobId: 7508d00000TtU9WAAV, Job Complete
21:32:40: BulkAPIv2Insert BACKGROUND completed successfully
-----------------------------------------------------------------
```

BulkAPIv2Update

Create a load table Example (WAIT)

This example creates a load table called "Account_BulkAPIv2_Wait_Update". The Id and Error columns are mandatory. Generally, an update payload would be built up from querying or working with prior replicated data for the affected Object by using ss_Replica / ss_Delta. All load operations require the Error column, whether success or failure, to guide and inform you.

```
drop table if exists Account_BulkAPIv2_Wait_Update
select top 100
convert(nchar(18),Id) as Id
,convert(nvarchar(255),null) as Error
,Id as AccountNumber
,AccountNumber as AccountNumber_Orig
into Account_BulkAPIv2_Wait_Update
from Account
where AccountNumber not like '0%'
order by createddate desc
```

Running the example

```
exec ss_Loader 'BulkAPIv2Update', 'DEMO', 'Account_BulkAPIv2_Wait_Update','WAIT'
```

```
SQL-SALES BulkAPIv2Update run date: 2023-12-09 ------------------
21:34:15: Using Env|Schema: DEMO|dbo
21:34:15: Starting Loader for Account batchsize 10000
21:34:15: SSId added to Account_BulkAPIv2_Wait_Update
21:34:18: Connection method BULK & BULK API
21:34:18: Bulk API method WAIT
21:34:18: Columns checked against Salesforce metadata
21:34:18: Starting load for Account_BulkAPIv2_Wait_Update
21:34:32: JobId: 7508d00000TtUCQAA3
21:34:33: Excluded: AccountNumber_Orig is not available on object Account
21:34:33: Load complete: Success:100 Failure:0
----------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

Ss_BulkAPILog table

The Job Id is also preserved in the Ss_BulkAPILog table, written on each submission.

Checking the load table

The success or failure errors for each row will be automatically written back to the Error column as with the SOAP API Update operation.

```
select * from Account_BulkAPIv2_Wait_Update
```

sults Messages

Id	Error	AccountNumber	AccountNumber_Orig	SSId
0018d00000rSErJAAW	Success	0018d00000rSErJAAW	20230921av1__0018d00000rRRHUAA4	1
0018d00000rSErIAAW	Success	0018d00000rSErIAAW	20230921av1__0018d00000rRRHtAAO	2
0018d00000rSErHAAW	Success	0018d00000rSErHAAW	20230921av1__0018d00000rRRHTAA4	3
0018d00000rSErGAAW	Success	0018d00000rSErGAAW	20230921av1__0018d00000rRRHsAAO	4
0018d00000rSErFAAW	Success	0018d00000rSErFAAW	20230921av1__0018d00000rRRHSAA4	5

Create a load table Example (BACK)

This example creates a load table called "Account_BulkAPIv2_BACK_Update". The Id and Error columns are mandatory. All load operations require the Error column, whether success or failure, to guide and inform you.

```
exec ss_Delta 'DEMO', 'Account'

drop table if exists Account_BulkAPIv2_BACK_Update
select top 100
convert(nchar(18),Id) as Id
,convert(nvarchar(255),null) as Error
,Id as AccountNumber
,AccountNumber as AccountNumber_Orig
into Account_BulkAPIv2_BACK_Update
from Account
where AccountNumber not like '0%'
order by createddate desc
```

Running the example (Step 1)

```
exec ss_Loader 'BulkAPIv2Update', 'DEMO', 'Account_BulkAPIv2_BACK_Update','BACK'
```

```
SQL-SALES BulkAPIv2Update run date: 2023-12-09 ------------------
21:37:02: Using Env|Schema: DEMO|dbo
21:37:03: Starting Loader for Account batchsize 10000
21:37:03: SSId added to Account_BulkAPIv2_BACK_Update
21:37:05: Connection method BULK & BULK API
21:37:05: Bulk API method BACK
21:37:05: Columns checked against Salesforce metadata
21:37:06: Starting load for Account_BulkAPIv2_BACK_Update
21:37:09: JobId: 7508d00000TtUCpAAN
21:37:09: BulkAPIv2Update BACKGROUND completed successfully
----------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

Note unlike the WAIT method, simply running BACK will not populate Error data to your load table, see the next set of instructions for how to do this, however the Batch Id(s) are included in the output dump for information purposes

Ss_BulkAPILog table

The Job Id is also preserved in the Ss_BulkAPILog table, written on each submission.

Running the example (Step 2 Option 1)

At any time after you have run the initial BACK, you can retrieve processed rows to your load table by reverting back to using the WAIT method. This is achieved by passing in the known Job Id into the @Special2 input parameter.

Note, when using "Option 1" if you have attempted to return processed rows "too soon" and some rows for a given Batch or Batches are not yet processed by Salesforce, SQL Sales will not be able to return an Error value hence caution should be exercised and for you to check your load table.

Alternatively, you can use "Option 2" to check the status of your Job by submitting a followup BACK request, alongside your known Job Id. This will instruct SQL Sales to check all related Batches and return the status of the Job Id. When the Job is Closed, no further processing will occur by Salesforce and you can now run with WAIT to return all Id and Error values.

```
exec ss_Loader 'BulkAPIv2Update', 'DEMO', 'Account_BulkAPIv2_BACK_Update','JOB:WAIT','
7508d00000TtUCpAAN'
```

```
SQL-SALES BulkAPIv2Update run date: 2023-12-09 ------------------
21:38:53: Using Env|Schema: DEMO|dbo
21:38:53: Starting Loader for Account batchsize 10000
21:38:53: SSId added to Account_BulkAPIv2_BACK_Update
21:38:56: Connection method BULK & BULK API
21:38:56: Bulk API method JOB:WAIT Job = 7508d00000TtUCpAAN
21:38:56: Columns checked against Salesforce metadata
21:38:57: Starting load for Account_BulkAPIv2_BACK_Update
21:39:00: JobId: 7508d00000TtUCpAAN, Job Complete
21:39:00: Excluded: AccountNumber_Orig is not available on object Account
21:39:00: Load complete: Success:100 Failure:0
------------------------------------------------------------------
```

Checking the load table

The success or failure errors for each row will be automatically written back to the Error column as with the SOAP API Update operation.

```
select * from Account_BulkAPIv2_BACK_Update
```

esults Messages

Id	Error	AccountNumber	AccountNumber_Orig	SSId
0018d00000rSEy2AAG	Success	0018d00000rSEy2AAG	DEMO__PREFIXCD451796	1
0018d00000rSEy3AAG	Success	0018d00000rSEy3AAG	DEMO__PREFIXCD656092	2
0018d00000rSEy4AAG	Success	0018d00000rSEy4AAG	DEMO__PREFIXCD439877	3
0018d00000rSEy5AAG	Success	0018d00000rSEy5AAG	DEMO__PREFIXCD355118	4
0018d00000rSEy6AAG	Success	0018d00000rSEy6AAG	DEMO__PREFIXCC947211	5

Running the example (Step 2 Option 2)

You can keep submitting with BACK and the known Job Id until the Status shows that the Job has Closed. Once the Job is Closed you can run as with Option 1.

```
exec ss_Loader 'BulkAPIv2Update', 'DEMO',
'Account_BulkAPIv2_BACK_Update','JOB:BACK','7508d00000TtUCpAAN'
```

```
SQL-SALES BulkAPIv2Update run date: 2023-12-09 ------------------
21:41:16: Using Env|Schema: DEMO|dbo
21:41:16: Starting Loader for Account batchsize 10000
21:41:16: SSId added to Account_BulkAPIv2_BACK_Update
21:41:19: Connection method BULK & BULK API
21:41:19: Bulk API method JOB:BACK Job = 7508d00000TtUCpAAN
21:41:19: Columns checked against Salesforce metadata
21:41:20: Starting load for Account_BulkAPIv2_BACK_Update
21:41:24: JobId: 7508d00000TtUCpAAN, Job Complete
21:41:24: BulkAPIv2Update BACKGROUND completed successfully
----------------------------------------------------------------------
```

BulkAPIv2Delete

Create a load table Example (WAIT)

This example creates a load table called "Account_BulkAPIv2_Wait_Delete". The Id and Error columns are mandatory. Generally, a delete payload would be built up from querying or working with prior replicated data for the affected Object by using ss_Replica / ss_Delta. All load operations require the Error column, whether success or failure, to guide and inform you.

```
exec ss_Delta 'DEMO', 'Account'

drop table if exists Account_BulkAPIv2_Wait_Delete
select top 100
convert(nchar(18),Id) as Id
,convert(nvarchar(255),null) as Error
,Name as Name_Info
into Account_BulkAPIv2_Wait_Delete
from Account
order by createddate desc
```

Running the example

```
exec ss_Loader 'BulkAPIv2Delete', 'DEMO', 'Account_BulkAPIv2_Wait_Delete','WAIT'
```

```
SQL-SALES BulkAPIv2Delete run date: 2023-12-09 ------------------
21:42:25: Using Env|Schema: DEMO|dbo
21:42:25: Starting Loader for Account batchsize 10000
21:42:25: SSId added to Account_BulkAPIv2_Wait_Delete
21:42:27: Connection method BULK & BULK API
21:42:27: Bulk API method WAIT
21:42:27: Columns checked against Salesforce metadata
21:42:28: Starting load for Account_BulkAPIv2_Wait_Delete
21:42:42: JobId: 7508d00000TtRuMAAV
21:42:43: Load complete: Success:100 Failure:0
----------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

Ss_BulkAPILog table

The Job Id is also preserved in the Ss_BulkAPILog table, written on each submission.

Checking the load table

The success or failure errors for each row will be automatically written back to the Error column as with the SOAP API Delete operation.

Create a load table Example (BACK)

This example creates a load table called "Account_BulkAPIv1_BACK_Serial_Delete". The Id and Error columns are mandatory. All load operations require the Error column, whether success or failure, to guide and inform you.

```
exec ss_Delta 'DEMO', 'Account'

drop table if exists Account_BulkAPIv2_BACK_Delete
select top 100
convert(nchar(18),Id) as Id
,convert(nvarchar(255),null) as Error
,Name as Name_Info
into Account_BulkAPIv2_BACK_Delete
from Account
order by createddate desc
```

Running the example (Step 1)

```
exec ss_Loader 'BulkAPIv2Delete', 'DEMO', 'Account_BulkAPIv2_BACK_Delete','BACK'
```

```
SQL-SALES BulkAPIv2Delete run date: 2023-12-09 ------------------
21:44:33: Using Env|Schema: DEMO|dbo
21:44:33: Starting Loader for Account batchsize 10000
21:44:33: SSId added to Account_BulkAPIv2_BACK_Delete
21:44:36: Connection method BULK & BULK API
21:44:36: Bulk API method BACK
21:44:36: Columns checked against Salesforce metadata
21:44:36: Starting load for Account_BulkAPIv2_BACK_Delete
21:44:39: JobId: 7508d00000TtUDOAA3
21:44:40: BulkAPIv2Delete BACKGROUND completed successfully
----------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

Note unlike the WAIT method, simply running BACK will not populate the _Batch table nor write back Error data to your load table, see the next set of instructions for how to do this, however the Batch Id(s) are included in the output dump for information purposes

Ss_BulkAPILog table

The Job Id is also preserved in the Ss_BulkAPILog table, written on each submission.

Running the example (Step 2 Option 1)

At any time after you have run the initial BACK, you can retrieve processed rows to your load table by reverting back to using the WAIT method. This is achieved by passing in the known Job Id into the @Special2 input parameter.

Note, when using "Option 1" if you have attempted to return processed rows "too soon" and some rows for a given Batch or Batches are not yet processed by Salesforce, SQL Sales will not be able to return an Error value hence caution should be exercised and for you to check your load table.

Alternatively, you can use "Option 2" to check the status of your Job by submitting a followup BACK request, alongside your known Job Id. This will instruct SQL Sales to check all related Batches and return the status of the Job Id. When the Job is Closed, no further processing will occur by Salesforce and you can now run with WAIT to return all Id and Error values.

```
exec ss_Loader 'BulkAPIv2Delete', 'DEMO',
'Account_BulkAPIv2_BACK_Delete','JOB:WAIT','7508d00000TtUDOAA3'
```

```
SQL-SALES BulkAPIv2Delete run date: 2023-12-09 ------------------
21:46:28: Using Env|Schema: DEMO|dbo
21:46:28: Starting Loader for Account batchsize 10000
21:46:28: SSId added to Account_BulkAPIv2_BACK_Delete
21:46:31: Connection method BULK & BULK API
21:46:31: Bulk API method JOB:WAIT Job = 7508d00000TtUDOAA3
21:46:31: Columns checked against Salesforce metadata
21:46:31: Starting load for Account_BulkAPIv2_BACK_Delete
21:46:35: JobId: 7508d00000TtUDOAA3, Job Complete
21:46:35: Load complete: Success:100 Failure:0
------------------------------------------------------------------
```

Checking the load table

The success or failure errors for each row will be automatically written back to the Error column as with the SOAP API Delete operation.

```
select * from Account_BulkAPIv2_BACK_Delete
```

sults Messages

Id	Error	Name_Info	SSId
0018d00000rSEw1AAG	Success	DEMO__PREFIXv2New Name, Upsert Demo 123	1
0018d00000rSEw2AAG	Success	DEMO__PREFIXv2New Name, Upsert Demo 789	2
0018d00000rSEw3AAG	Success	DEMO__PREFIXv2AAAAAA46	3
0018d00000rSEw4AAG	Success	DEMO__PREFIXv2NameChange7gAAA	4
0018d00000rSEw5AAG	Success	DEMO__PREFIXv2NameChange7gAAA	5

Running the example (Step 2 Option 2)

You can keep submitting with BACK and the known Job Id until the Status shows that the Job has Closed. Once the Job is Closed you can run as with Option 1.

```
exec ss_Loader 'BulkAPIv2Delete', 'DEMO',
'Account_BulkAPIv2_BACK_Delete','JOB:BACK','7508d00000TtUDOAA3'
```

```
SQL-SALES BulkAPIv2Delete run date: 2023-12-09 ------------------
21:47:59: Using Env|Schema: DEMO|dbo
21:47:59: Starting Loader for Account batchsize 10000
21:47:59: SSId added to Account_BulkAPIv2_BACK_Delete
21:48:02: Connection method BULK & BULK API
21:48:02: Bulk API method JOB:BACK Job = 7508d00000TtUDOAA3
21:48:02: Columns checked against Salesforce metadata
21:48:03: Starting load for Account_BulkAPIv2_BACK_Delete
21:48:06: JobId: 7508d00000TtUDOAA3, Job Complete
21:48:06: BulkAPIv2Delete BACKGROUND completed successfully
----------------------------------------------------------------
```

BulkAPIv2Harddelete

Create a load table Example (WAIT)

This example creates a load table called "Account_BulkAPIv2_Wait_Harddelete". The Id and Error columns are mandatory. Generally, an undelete payload would be built up from querying or working with prior deleted data for the affected Object. All load operations require the Error column, whether success or failure, to guide and inform you.

```
exec ss_Delta 'DEMO', 'Account'

drop table if exists Account_BulkAPIv2_Wait_Harddelete
select top 100
convert(nchar(18),Id) as Id
,convert(nvarchar(255),null) as Error
,Name as Name_Info
into Account_BulkAPIv2_Wait_Harddelete
from Account
order by createddate desc
```

Running the example

```
exec ss_Loader 'BulkAPIv2Harddelete', 'DEMO', 'Account_BulkAPIv2_Wait_Harddelete','WAIT'
```

```
SQL-SALES BulkAPIv2Harddelete run date: 2023-12-09 --------------
21:49:27: Using Env|Schema: DEMO|dbo
21:49:27: Starting Loader for Account batchsize 10000
21:49:27: SSId added to Account_BulkAPIv2_Wait_Harddelete
21:49:30: Connection method BULK & BULK API
21:49:30: Bulk API method WAIT
21:49:30: Columns checked against Salesforce metadata
21:49:31: Starting load for Account_BulkAPIv2_Wait_Harddelete
21:49:32: Failed to create job. Status code: 400, which in V2 of the Bulk API can mean your User does not have Hard delete permissions
-----------------------------------------------------------------
```

Note, by default a sys admin user will not have permission to run Harddelete, you will need a special profile created (in fact cloned) from a system administrator and Dard delete enabled on it, without this initial preparation you will encounter the above error message

```
exec ss_Loader 'BulkAPIv1Harddelete', 'HARD',
'Account_BulkAPIv1_Wait_Serial_Harddelete','WAIT:SERIAL'
```

A special Envionment has been setup in this Demo, called “HARD” (which uses a different username, with a Harddelete profile enabled

```
SQL-SALES BulkAPIv2Harddelete run date: 2023-12-09 --------------
21:50:36: Using Env|Schema: HARD|dbo
21:50:36: Starting Loader for Account batchsize 10000
```

```
21:50:36: SSId added to Account_BulkAPIv2_Wait_Harddelete
21:50:39: Connection method BULK & BULK API
21:50:39: Bulk API method WAIT
21:50:39: Columns checked against Salesforce metadata
21:50:39: Starting load for Account_BulkAPIv2_Wait_Harddelete
21:50:54: JobId: 7508d00000TtU9dAAF
21:50:55: Load complete: Success:100 Failure:0
-----------------------------------------------------------------
```

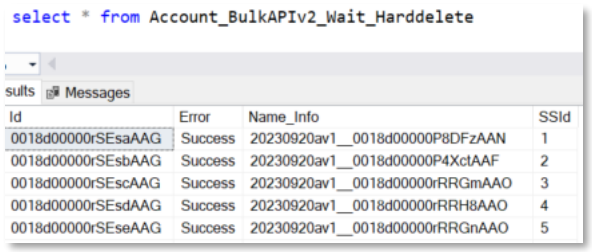
Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

Ss_BulkAPILog table

The Job Id is also preserved in the Ss_BulkAPILog table, written on each submission.

Checking the load table

The success or failure errors for each row will be automatically written back to the Error column.



```
select * from Account_BulkAPIv2_Wait_Harddelete
```

sults | Messages

Id	Error	Name_Info	SSId
0018d00000rSEsaAAG	Success	20230920av1__0018d00000P8DFzAAN	1
0018d00000rSEsbAAG	Success	20230920av1__0018d00000P4XctAAF	2
0018d00000rSEscAAG	Success	20230920av1__0018d00000rRRGmAAO	3
0018d00000rSEsdAAG	Success	20230920av1__0018d00000rRRH8AAO	4
0018d00000rSEseAAG	Success	20230920av1__0018d00000rRRGnAAO	5

Create a load table Example (BACK)

This example creates a load table called "Account_BulkAPIv2_BACK_Example_Harddelete". The Id and Error columns are mandatory. All load operations require the Error column, whether success or failure, to guide and inform you.

```
exec ss_Delta 'DEMO', 'Account'

drop table if exists Account_BulkAPIv2_BACK_Harddelete
select top 100
convert(nchar(18),Id) as Id
,convert(nvarchar(255),null) as Error
,Name as Name_Info
into Account_BulkAPIv2_BACK_Harddelete
from Account
order by createddate desc
```

Running the example (Step 1)

```
exec ss_Loader 'BulkAPIv2Harddelete', 'HARD', 'Account_BulkAPIv2_BACK_Harddelete','BACK'
```

```
SQL-SALES BulkAPIv2Harddelete run date: 2023-12-09 --------------
21:53:22: Using Env|Schema: HARD|dbo
21:53:22: Starting Loader for Account batchsize 10000
21:53:22: SSId added to Account_BulkAPIv2_BACK_Harddelete
21:53:25: Connection method BULK & BULK API
21:53:25: Bulk API method BACK
21:53:25: Columns checked against Salesforce metadata
21:53:25: Starting load for Account_BulkAPIv2_BACK_Harddelete
21:53:29: JobId: 7508d00000TtU3tAAF
21:53:29: BulkAPIv2Harddelete BACKGROUND completed successfully
---------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

Note unlike the WAIT method, simply running BACK will not populate the _Batch table nor write back Error data to your load table, see the next set of instructions for how to do this, however the Batch Id(s) are included in the output dump for information purposes

Ss_BulkAPILog table

The Job Id is also preserved in the Ss_BulkAPILog table, written on each submission.

Running the example (Step 2 Option 1)

At any time after you have run the initial BACK, you can retrieve processed rows to your load table by reverting back to using the WAIT method (either in SERIAL or PARALLEL mode). This is achieved by passing in the known Job Id into the @Special2 input parameter.

Note, when using “Option 1” if you have attempted to return processed rows “too soon” and some rows for a given Batch or Batches are not yet processed by Salesforce, SQL Sales will not be able to return an Error value hence caution should be exercised and for you to check your load table.

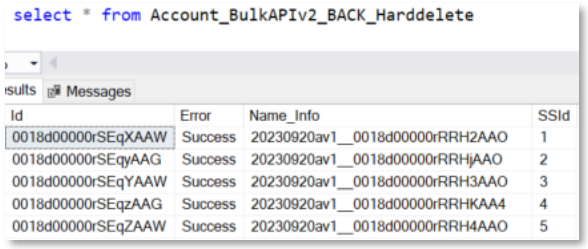
Alternatively, you can use “Option 2” to check the status of your Job by submitting a followup BACK request, alongside your known Job Id. This will instruct SQL Sales to check all related Batches and return the status of the Job Id. When the Job is Closed, no further processing will occur by Salesforce and you can now run with WAIT to return all Id and Error values.

```
exec ss_Loader 'BulkAPIv2Harddelete', 'HARD',
'Account_BulkAPIv2_BACK_Harddelete','JOB:WAIT','7508d00000TtU3tAAF'
```

```
SQL-SALES BulkAPIv2Harddelete run date: 2023-12-09 --------------
21:54:40: Using Env|Schema: HARD|dbo
21:54:40: Starting Loader for Account batchsize 10000
21:54:40: SSId added to Account_BulkAPIv2_BACK_Harddelete
21:54:43: Connection method BULK & BULK API
21:54:43: Bulk API method JOB:WAIT Job = 7508d00000TtU3tAAF
21:54:43: Columns checked against Salesforce metadata
21:54:43: Starting load for Account_BulkAPIv2_BACK_Harddelete
21:54:47: JobId: 7508d00000TtU3tAAF, Job Complete
21:54:47: Load complete: Success:100 Failure:0
----------------------------------------------------------------
```

Checking the load table

The success or failure errors for each row will be automatically written back to the Error column.



```
select * from Account_BulkAPIv2_BACK_Harddelete
```

sults | Messages

Id	Error	Name_Info	SSId
0018d00000rSEqXAAW	Success	20230920av1__0018d00000rRRH2AAO	1
0018d00000rSEqyAAG	Success	20230920av1__0018d00000rRRHjAAO	2
0018d00000rSEqYAAW	Success	20230920av1__0018d00000rRRH3AAO	3
0018d00000rSEqzAAG	Success	20230920av1__0018d00000rRRHKAA4	4
0018d00000rSEqZAAW	Success	20230920av1__0018d00000rRRH4AAO	5

Running the example (Step 2 Option 2)

You can keep submitting with BACK and the known Job Id until the Status shows that the Job has Closed. Once the Job is Closed you can run as with Option 1.

```
exec ss_Loader 'BulkAPIv2Harddelete', 'HARD',
'Account_BulkAPIv2_BACK_Harddelete','JOB:BACK','7508d00000TtU3tAAF'
```

```
SQL-SALES BulkAPIv2Harddelete run date: 2023-12-09 --------------
21:56:34: Using Env|Schema: HARD|dbo
21:56:34: Starting Loader for Account batchsize 10000
21:56:34: SSId added to Account_BulkAPIv2_BACK_Harddelete
21:56:36: Connection method BULK & BULK API
21:56:36: Bulk API method JOB:BACK Job = 7508d00000TtU3tAAF
21:56:36: Columns checked against Salesforce metadata
21:56:37: Starting load for Account_BulkAPIv2_BACK_Harddelete
21:56:40: JobId: 7508d00000TtU3tAAF, Job Complete
21:56:40: BulkAPIv2Harddelete BACKGROUND completed successfully
----------------------------------------------------------------
```

BulkAPIv2Upsert

As with the SOAP API Upsert, working with the Salesfore Bulk API v2 is a little different to the other operations. BulkAPIv2Update, BulkAPIv2Delete and BulkAPIv2Harddelete work from the provided Id, with regards what records to operate against. Insert merely creates new records and passes the new Id back.

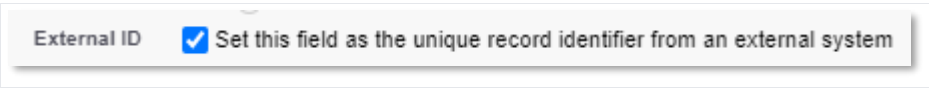
Whereas BulkAPIv2Upsert, as the name suggests, works primarily off a provided External Id for the given object being upserted against.

If a match is found in that Salesforce object for the value being passed in, then the Operation will update the provided fields in the table payload to the matched record.

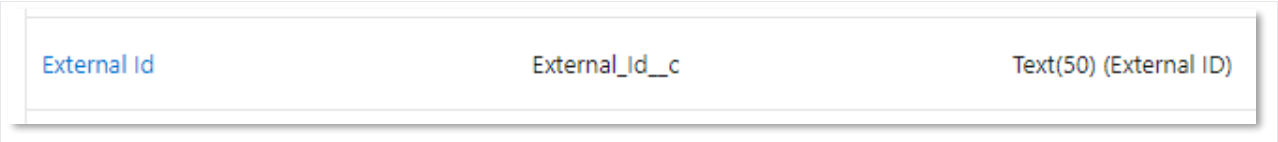
If a match is not found, the Upsert operation will insert a new record, using the provided fields in the table payload. All load operations require the Error column, whether success or failure, to guide and inform you.

External Id

Upsert will only work against an External data type field. This is a special setting for a field in Salesforce:



For this example, the field "External_Id__c" has been added to the Account object, note the special indicator "(External ID)". If you intended External Id does not have this, it is likely not actually setup as an External Id, no matter what the field name is.



SQL Sales will inform if it is not truly an External Id, this is demonstrated in the following examples.

Create a load table Example (WAIT)

This example creates a load table called "Account_BulkAPIv2_Wait_Upsert".

```
drop table if exists Account_BulkAPIv2_Wait_Upsert
select top 33
convert(nchar(18),null) as Id
,convert(nvarchar(255),null) as Error
,Name
,'UPDATE_TEST_' + Id as AccountNumber
,External_Id__c
,AccountNumber as AccountNumber_Orig
into Account_BulkAPIv2_Wait_Upsert
from Account
where AccountNumber not like '0%'
and External_Id__c is not null
order by createddate desc

insert Account_BulkAPIv2_Wait_Upsert
(Id
,Error
,Name
,AccountNumber
,External_Id__c
,AccountNumber_Orig)

select top 33
convert(nchar(18),null) --Id
,convert(nvarchar(255),null) --Error
,Name
,'REJECT_TEST_' + Id --AccountNumber
,null --External_Id__c
,null --AccountNumber_Orig
from Account
where AccountNumber not like '0%'
and External_Id__c is not null
order by createddate desc

insert Account_BulkAPIv2_Wait_Upsert
(Id
,Error
,Name
,AccountNumber
,External_Id__c
,AccountNumber_Orig)

select top 33
convert(nchar(18),null) --Id
,convert(nvarchar(255),null) --Error
,Name
,'INSERT_TEST_' + Id --AccountNumber
,'INSERT_TEST_' + Id --AExternal_Id__c
,null --AccountNumber_Orig
from Account
where AccountNumber not like '0%'
and External_Id__c is not null
order by createddate desc
```

Check the Payload

Note, those records with a matched (to Salesforce) value in External_Id__c will result in an update, those with a new value but no match to Salesforce will result in an Insert, whereas those with no

value in External_Id__c will be rejected (as External_Id__c, being the specified External Id field for the BulkAPIv1Upsert operation, es expected to be populated).

```
select * from Account_BulkAPIv2_Wait_Upsert
```

sults Messages

Id	Error	Name	AccountNumber	External_Id__c	AccountNumber_Orig
NULL	NU...	20230920av1__0018d00000P4XctAAF	UPDATE_TEST_0018d00000rSEkBAAW	Test0018d00000P4XctAAF	20230921av1__0018d00000P4XctAAF
NULL	NU...	20230920av1__0018d00000P4XctAAF	INSERT_TEST_0018d00000rSEkBAAW	INSERT_TEST_0018d00000rSEkBA...	NULL
NULL	NU...	20230920av1__0018d00000P4XctAAF	REJECT_TEST_0018d00000rSEkBAAW	NULL	NULL
NULL	NU...	20230920av1__0018d00000P8DFzAAN	REJECT_TEST_0018d00000rSF2dAAG	NULL	NULL
NULL	NU...	20230920av1__0018d00000P8DFzAAN	INSERT_TEST_0018d00000rSF2dAAG	INSERT_TEST_0018d00000rSF2dAA...	NULL

Running the example (incorrect)

```
exec ss_Loader 'BulkAPIv2Upsert', 'DEMO', 'Account_BulkAPIv2_Wait_Upsert','WAIT'
```

```
SQL-SALES BulkAPIv2Upsert run date: 2023-12-09 ------------------
22:02:11: Using Env|Schema: DEMO|dbo
22:02:11: Provided Upsert Operation value not in the correct format, for a hypothetical field
called "External_Id__c"
22:02:11: set (in Salesforce) as an External Id (this is a field setting)
22:02:11: the expected input value for the standard web services API is:
Upsert:XId=External_Id__c
22:02:11: or BulkAPIv1Upsert:XId=External_Id__c for the bulk API v1
22:02:11: or BulkAPIv2Upsert:XId=External_Id__c for the bulk API v2
22:02:11: with a defined batchsize (for example 100 or 1000 respectively) these would be:
22:02:11: Upsert(100):XId=External_Id__c | BulkAPIv1Upsert(10000):XId=External_Id__c |
BulkAPIv2Upsert(2000):XId=External_Id__c
22:02:11: Note, the provided External Id is validated directly against Salesforce Account
22:02:11: prior to the run commencing to check it is actually defined as an External Id.
-----------------------------------------------------------------
```

Note, the format provided in the example is incorrect, observe the supporting help text

Running the example (correct)

```
exec ss_Loader 'BulkAPIv2Upsert:XId=External_Id__c', 'DEMO',
'Account_BulkAPIv2_Wait_Upsert','WAIT'
```

```
SQL-SALES BulkAPIv2Upsert:XId=External_Id__c run date: 2023-12-09
22:02:55: Using Env|Schema: DEMO|dbo
22:02:58: Starting Loader for Account batchsize 10000
22:02:58: SSId added to Account_BulkAPIv2_Wait_Upsert
22:03:01: Connection method BULK & BULK API
22:03:01: Bulk API method WAIT
22:03:01: Columns checked against Salesforce metadata
22:03:01: Starting load for Account_BulkAPIv2_Wait_Upsert
22:03:16: JobId: 7508d00000TtUJvAAN
22:03:16: Excluded: AccountNumber_Orig is not available on object Account
22:03:16: Load complete: Success:66 Failure:33
------------------------------------------------------------------
```

Note the Job Id for your submission is returned in the output for your reference, see also the log table ss_BulkAPILog

Ss_BulkAPILog table

The Job Id is also preserved in the Ss_BulkAPILog table, written on each submission.

Checking the load table

For Inserts and Version 2, no direct response of Ids and Error column values are written back to the Load table.

_Return helper table

The batch to which a given row has been allocated is logged in the helper table created in the run that is your load table + "_Return", using the above example it is:

Account_BulkAPIv2_Wait_Upsert_Return

For Version 2, this provides the created Id and/or Error reason for information purposes, but the rows do not tire back to your load table.

TECHNICAL OVERVIEW

SQL-Sales is an application installed on a SQL Server. It receives instructions from SQL Server Management Studio (SSMS) stored procedures to read and write to Salesforce. It comprises 3 executables:

SQLSalesConfig.exe

- Configuration tool (with UI), where connections to Salesforce are defined
- Connections and data requests are via basic Username+Password+Security token (SOAP or REST) as well as OAuth2.0 (JWT, via REST)

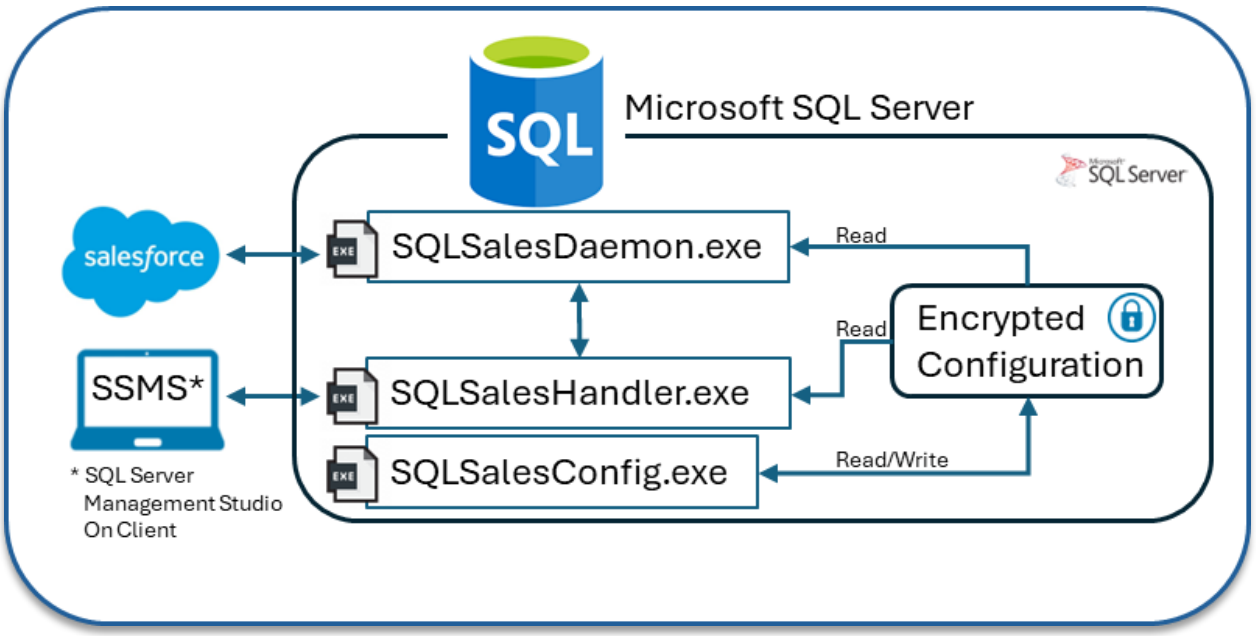
SQLSalesDaemon.exe

- Constantly running Daemon process, connecting to Salesforce on demand, according to instructions from the end user (in SSMS) via the Handler

SQLSalesHandler.exe

- Handles requests from SQL Server Management Studio (SSMS), which are passed to the Daemon

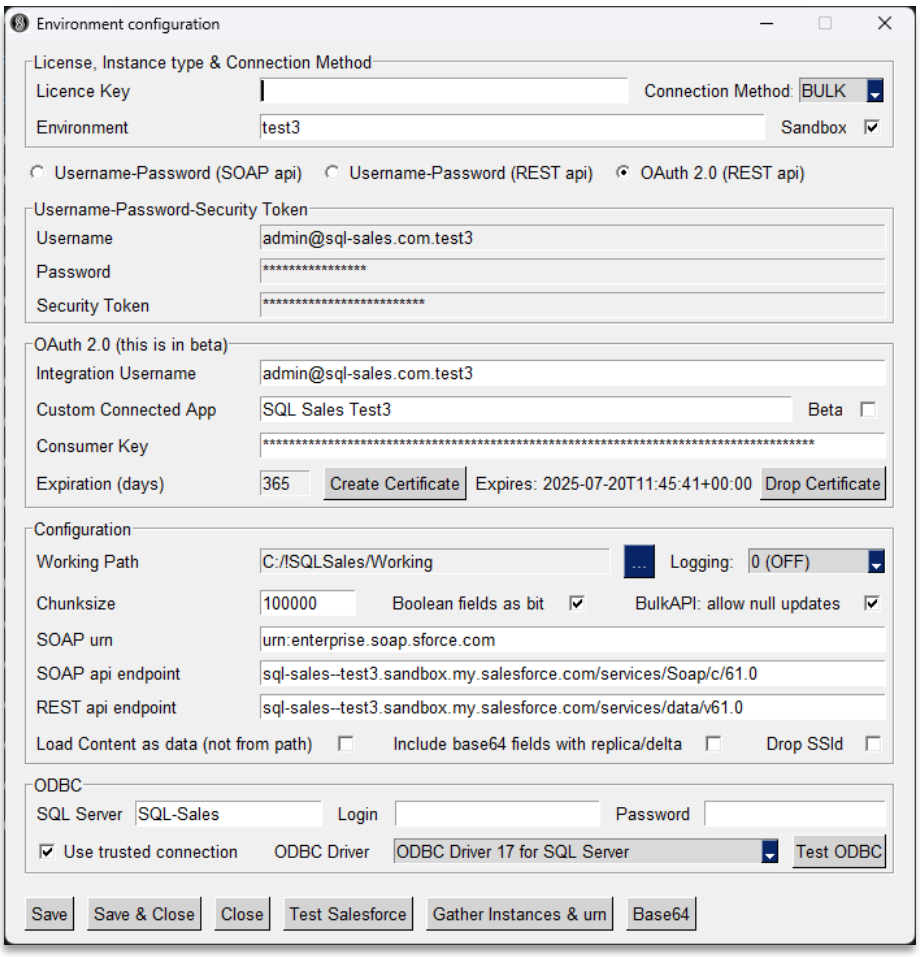
The installation is outlined below

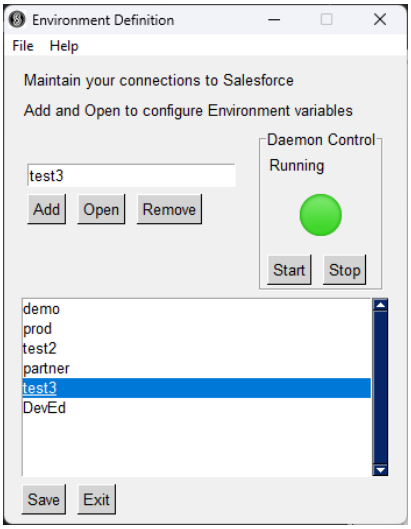


SQL-Sales

SQL-Sales Config

This is where credentials and customisations to how data is processed, per given Salesforce instance, are defined. As shown, the connection methods are via traditional Username+Password+Security token (SOAP or REST) as well as OAuth2.0 (JWT, via REST).





Shown above top right are the Environments defined for the installation and a visual indicator of whether the Daemon is running/stopped – as well as the ability to start the Daemon (although running a SQL-Sales stored procedure in SSMS for an enabled Database will auto-start the Daemon). Top left is an example (in this case for a sandbox called "demo").

The process for storing configuration credential and attributes securely involves a series of steps designed to protect the data from unauthorised access by encrypting it before saving it to a file. This ensures that even if someone gains access to the storage file (unlikely on a SQL Server), they will not be able to understand or use the stored data without the correct key.

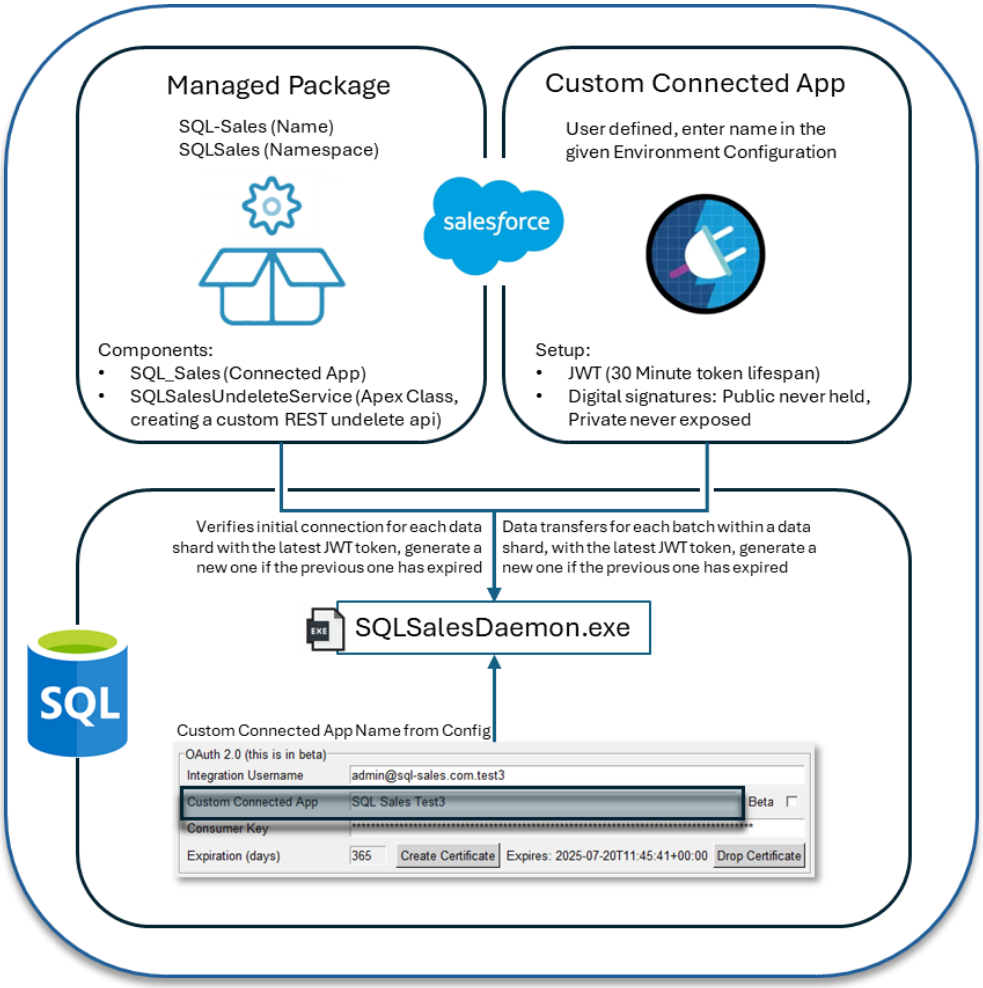
SQL-Sales

Note, the configuration file storage location can be fully configured by the SQL Server user. Typically this will be saved directly on the SQL Server itself (i.e. with the inherent protection of being on the root drives of the SQL Server). Alternatively, the storage location can be a network drive for example to which only the user of the SQL-Sales Config tool has access to.

OAuth 2.0 Setup

Whilst the traditional Username+Password+Security token is supported and requires no Salesforce setup other than the Managed Package, for enhanced security, an OAuth 2.0 connection option is offered which requires minimal Salesforce configuration. SQL-Sales OAuth 2.0 will intentionally not work without an additional user-created connected app, for the given Salesforce instance and SQL-Sales configuration installation/SQL Server. For maximum security each SQL-Sales config installation (on potentially multiple SQL Servers) connecting to the same Salesforce instance will need their own dedicated Custom Connected App setting up.

The setup process is documented here



In summary, the steps for creating a custom connected app are:

1. In Salesforce. Create the new Connected App in the given Salesforce instance
    a. Enable OAuth
    b. Use digital signatures
    c. Selected OAuth Scopes
        i. Manage user data via APIs (api)
        ii. Perform requests at any time (refresh_token, offline_access)
    d. Issue JSON Web Token (JWT)-based access tokens for named users
    e. OAuth Policies
        i. Admin approved users are pre-authorized
        ii. Issue JSON Web Token (JWT)-based access tokens (ticked)
        iii. Token Timeout (30 Minutes)
        iv. Enforce IP restrictions (Refresh token is valid until revoked)
    f. Manage Profiles
        i. Selected Profile relevant for the nominated integration user
    g. Manage Consumer Details
        i. Copy the Consumer Key
2. In SQL-Sales Configuration
    a. Enter Integration Username (the nominated integration user)
    b. Enter Custom Connected App (the name of the connected app created in step 1 above)
    c. Enter (paste) the Consumer Key from step 1 above
    d. Enter Expiration (days) – maximum permitted is 365
    e. Click "Create Certificate" this will automatically securely store the private key within the SQL-Sales Config storage mechanism and pass the public key to the clipboard, at no point (ever) does SQL-Sales save or store the public key
    f. User is wholly responsible for the creation and key vault storage (if relevant) of the resultant created .pem file where the key is pasted from the clipboard
3. In Salesforce. Return to the custom connected app created in Step 1
    a. Use digital signatures
        i. Choose File (of the .pem user created file from Step 2)
4. In SQL-Sales Configuration
    a. Click "Test Salesforce" – that's at, OAuth2.0 setup is complete, this is a double-pronged security model where a Manage Package (Managed connected App) is required, alongside a custom instance-installation specific connected app must be setup to work alongside.

installation Summary

A step by step guide

1. Install the SQL-Sales Managed Package, see here
2. Download installation zip SQL-Sales.zip from here. The zip contains:
    a. SQL-Sales read me.txt
    b. SQL-Sales Guide.pdf
    c. SQLSalesInstaller.exe
3. Extract SQLSalesInstaller.exe from the zip and run, setup instructions here, this installs:
    a. SQLSalesConfig.exe
    b. SQLSalesHandler.exe
    c. SQLSalesDaemon.exe
    d. UninstallSQLSales.exe
    e. ss_EnableDatabase.sql
4. Enable your required SQL Server Database, instructions here
5. On the SQL Server, open "SQL-Sales Config"
    a. Create an Environment and configure as required
    b. Start the Daemon (optional, it will self-start on first use)
6. Create a custom Conncted App if required, see here
7. On either a client SQL Server Management Studio (SSMS) or from SSMS directly on the SQL Server, run the required stored procedure to read or write to Salesforce, depending on what you want to do, full online user documentation here, or standalone pdf here.

LICENCING ARRANGEMENTS

There are no limits on the use of SQL-Sales either commercially or personally, other than via a trial or paid-for license key. All SQL-Sales software components and libraries are detailed in this section in fulfilment of respective licencing controls, governing their use.

cryptography

Name: cryptography

Version: 39.0.0

Summary: cryptography is a package which provides cryptographic recipes and primitives to Python developers.

Home-page: https://github.com/pyca/cryptography

Author: The Python Cryptographic Authority and individual contributors

Author-email: cryptography-dev@python.org

License: (Apache-2.0 OR BSD-3-Clause) AND PSF-2.0

Requires: cffi

Licence Details:

https://github.com/pyca/cryptography/blob/main/LICENSE

https://github.com/pyca/cryptography/blob/main/LICENSE.BSD

see “BSD 3-Clause License” section in this document

https://github.com/pyca/cryptography/blob/main/LICENSE.APACHE

see “Apache 2.0” section in this document

This software is made available under the terms of *either* of the licenses found in LICENSE.APACHE or LICENSE.BSD. Contributions to cryptography are made under the terms of *both* these licenses.

psutil

Name: psutil

Version: 5.9.4

Summary: Cross-platform lib for process and system monitoring in Python.

Home-page: https://github.com/giampaolo/psutil

Author: Giampaolo Rodola

Author-email: g.rodola@gmail.com

License: BSD-3-Clause

Requires: n/a

Licence Details:

https://github.com/giampaolo/psutil/blob/master/LICENSE

see “BSD 3-Clause License” section in this document

giampaolo/psutil is licensed under the

BSD 3-Clause "New" or "Revised" License

A permissive license similar to the BSD 2-Clause License, but with a 3rd clause that prohibits others from using the name of the copyright holder or its contributors to promote derived products without written consent.

requests

Name: requests

Version: 2.28.2

Summary: Python HTTP for Humans.

Home-page: https://requests.readthedocs.io

Author: Kenneth Reitz

Author-email: me@kennethreitz.org

License: Apache 2.0

Requires: certifi, charset-normalizer, idna, urllib3

Required-by: requests-file, requests-toolbelt, simple-salesforce, zeep

Licence Details:

https://github.com/opentracing-contrib/python-requests/blob/master/LICENSE

see “Apache 2.0” section in this document

Copyright 2018 SignalFx, Inc. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License.

simple_salesforce

Name: simple-salesforce

Version: 1.12.3

Summary: A basic Salesforce.com REST API client.

Home-page: https://github.com/simple-salesforce/simple-salesforce

Author: Nick Catalano

Author-email: nickcatal@gmail.com

License: Apache 2.0

Requires: pyjwt, requests, zeep

Licence Details:

https://github.com/simple-salesforce/simple-salesforce/blob/master/LICENSE.txt

see “Apache 2.0” section in this document

Copyright 2012 New Organizing Institute Education Fund Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License.

sqlalchemy

Name: SQLAlchemy

Version: 2.0.13

Summary: Database Abstraction Library

Home-page: https://www.sqlalchemy.org

Author: Mike Bayer

Author-email: mike_mp@zzzcomputing.com

License: MIT

Requires: greenlet, typing-extensions

Licence Details:

https://opensource.org/license/mit/

see “MIT License” section in this document

opentracing-contrib/python-sqlalchemy is licensed under the

Apache License 2.0

A permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

pandas

Name: pandas

Version: 1.5.3

Summary: Powerful data structures for data analysis, time series, and statistics

Home-page: https://pandas.pydata.org

Author: The Pandas Development Team

Author-email: pandas-dev@python.org

License: BSD-3-Clause

Requires: numpy, numpy, python-dateutil, pytz

Licence Details:

https://github.com/pandas-dev/pandas/blob/main/LICENSE

see “BSD 3-Clause License” section in this document

pandas-dev/pandas is licensed under the

BSD 3-Clause "New" or "Revised" License

A permissive license similar to the BSD 2-Clause License, but with a 3rd clause that prohibits others from using the name of the copyright holder or its contributors to promote derived products without written consent.

pyodbc

Name: pyodbc

Version: 4.0.35

Summary: DB API Module for ODBC

Home-page: https://github.com/mkleehammer/pyodbc

Author:

Author-email:

License: MIT

Requires: n/a

Licence Details:

https://opensource.org/license/mit/

see “MIT License” section in this document

mkleehammer/pyodbc is licensed under the

MIT No Attribution

A short and simple permissive license with no conditions, not even requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

pysimplegui

Name: PySimpleGUI

Version: 4.60.4

Summary: Python GUIs for Humans. Launched in 2018. It's 2022 & PySimpleGUI is an ACTIVE & supported project. Super-simple to create custom GUI's. 325+ Demo programs & Cookbook for rapid start. Extensive documentation. Main docs at www.PySimpleGUI.org. Fun & your success are the focus. Examples using Machine Learning (GUI, OpenCV Integration), Rainmeter Style Desktop Widgets, Matplotlib + Pyplot, PIL support, add GUI to command line scripts, PDF & Image Viewers. Great for beginners & advanced GUI programmers.

Home-page: https://github.com/PySimpleGUI/PySimpleGUI

Author: PySimpleGUI

Author-email: PySimpleGUI@PySimpleGUI.org

License: GNU Lesser General Public License v3.0

Requires: n/a

Licence Details:

https://www.gnu.org/licenses/lgpl-3.0.en.html

see “GNU Lesser General Public License v3.0” section in this document

Permissions of this copyleft license are conditioned on making available complete source code of licensed works and modifications under the same license or the GNU GPLv3. Copyright and license notices must be preserved. Contributors provide an express grant of patent rights. However, a larger work using the licensed work through interfaces provided by the licensed work may be distributed under different terms and without source code for the larger work.

PIL

Name: PIL

Summary: Python Imaging Library

Author: Fredrik Lundh / Secret Labs AB

License: HPND

Requires: n/a

Licence Details:

see “HPND” section in this document

The Python Imaging Library (PIL) is

Pillow is the friendly PIL fork. It is

Like PIL, Pillow is licensed under the open source PIL Software License:

By obtaining, using, and/or copying this software and/or its associated documentation, you agree that you have read, understood, and will comply with the following terms and conditions:

Permission to use, copy, modify, and distribute this software and its associated documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appears in all copies, and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Secret Labs AB or the author not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

SECRET LABS AB AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL SECRET LABS AB OR THE AUTHOR BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

PSFL components

Licence Details:

License: PDF 2.0

Requires: n/a

https://spdx.org/licenses/PSF-2.0.html

see “PDF 2.0” section in this document

The following libraries are referenced as-is:

- datetime
- argparse
- base64
- calendar
- configparser
- csv
- gc
- hashlib
- io
- json
- logging
- os
- re
- signal
- socket
- subprocess
- sys
- tempfile
- textwrap
- threading
- time
- xml.etree.ElementTree
- ctypes
- uuid
- traceback

defusedxml

Name: defusedxml

Version: 0.7.1

Summary: defusedxml is a Python library that protects against XML-based attacks such as XML bomb and XML External Entity (XXE) attacks.

Home-page: https://github.com/tiran/defusedxml

Author: Christian Heimes

Author-email: christian@python.org

License: Python Software Foundation License

Requires: None

Licence Details: The defusedxml library is distributed under the Python Software Foundation License (PSF License). This license is permissive, similar to the BSD and Apache licenses, and it allows for free use, modification, and distribution, including in commercial products. The software is provided "as-is" without any warranties, so you must not claim any warranty or liability for the defusedxml library in your own distribution.

pyinstaller

Name: pyinstaller

Version: 5.9.0

Summary: PyInstaller bundles a Python application and all its dependencies into a single package.

Home-page: https://www.pyinstaller.org/

Author: Hartmut Goebel, Giovanni Bajo, David Vierra, David Cortesi, Martin Zibricky

Author-email:

License: GPLv2-or-later with a special exception which allows to use PyInstaller to build and distribute non-free programs (including commercial ones)

Requires: altgraph, pefile, pyinstaller-hooks-contrib, pywin32-ctypes, setuptools

Licence Details:

https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html

see “GPL 2.0” section in this document

PyInstaller is distributed under a dual-licensing scheme using both the GPL 2.0 License, with an exception that allows you to use it to build commercial products - listed below - and the Apache License, version 2.0, which only applies to a certain few files. To see which files the Apache license applies to, and to which the GPL applies, please see the COPYING.txt file which can be found in the root of the PyInstaller source repository.

A quick summary of the GPL license exceptions: You may use PyInstaller to bundle commercial applications out of your source code.

The executable bundles generated by PyInstaller from your source code can be shipped with whatever license you want, as long as it complies with the licenses of your dependencies.

You may modify PyInstaller for your own needs but changes to the PyInstaller source code fall under the terms of the GPL license. That is, if you distribute your modifications you must distribute them under GPL terms.

Nullsoft scriptable install system

Name: NSIS

Summary: NSIS is a professional open source system to create Windows installers

Home-page: https://nsis.sourceforge.io/Main_Page

Author: Amir Szekely

Author-email: kichik@gmail.com

Applicable licenses

All NSIS source code, plug-ins, documentation, examples, header files and graphics, with the exception of the compression modules and where otherwise noted, are licensed under the zlib/libpng license.

The zlib compression module for NSIS is licensed under the zlib/libpng license.

The bzip2 compression module for NSIS is licensed under the bzip2 license.

The lzma compression module for NSIS is licensed under the Common Public License version 1.0.

Copyright

Copyright (C) 1999-2024 Contributors

More detailed copyright information can be found in the individual source code files.

APACHE 2.0 LICENSE

Version 2.0, January 2004

https://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition,

"control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of

their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

You must give any other recipients of the Work or Derivative Works a copy of this License; and

You must cause any modified files to carry prominent notices stating that You changed the files; and

You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A

PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

BSD 3-Clause License

Copyright (c) Individual contributors.

All rights reserved.

Redistribution and use in source and binary forms, with or without

modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND

ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED

WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR

ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

MIT LICENSE

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

GNU Lesser General Public v3.0 License

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <https://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License.

"The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or

b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.

b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.

b) Accompany the Combined Work with a copy of the GNU GPL and this license document.

c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.

d) Do one of the following:

0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.

1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.

e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.

b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.

3. In the event Licensee prepares a derivative work that is based on or incorporates Python or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python.

4. PSF is making Python available to Licensee on an "AS IS" basis. PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT
LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE
OF PYTHON WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between PSF and Licensee. This License Agreement does not grant permission to use PSF trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.

8. By copying, installing or otherwise using Python, Licensee agrees to be bound by the terms and conditions of this License Agreement.

Standard License Header

There is no standard license header for the license

GPL 2.0 LICENSE

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Preamble The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program",

below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so,

and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software

Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

HPND LICENSE

By obtaining, using, and/or copying this software and/or its associated documentation, you agree that you have read, understood, and will comply with the following terms and conditions:

Permission to use, copy, modify and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appears in all copies, and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Secret Labs AB or the author not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

PDF 2.0 LICENSE

Full name

Python Software Foundation License 2.0

Short identifier

PSF-2.0

Other web pages for this license

https://opensource.org/licenses/Python-2.0 [no longer live]

Notes

This is the PSF-2.0 license, which is part of the complete Python license text, but also used independently by some projects.

PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee") accessing and otherwise using this software ("Python") in source or binary form and its associated documentation.

2. Subject to the terms and conditions of this License Agreement, PSF hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python alone or in any derivative version, provided, however, that PSF's License Agreement and PSF's notice of copyright , i.e., "Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006 Python Software Foundation All Rights Reserved" are retained in Python alone or in any derivative version prepared by Licensee.

zlib/libpng license

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

This notice may not be removed or altered from any source distribution.

bzip2 license

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Julian Seward, Cambridge, UK.

jseward@acm.org

Common Public License version 1.0

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS COMMON PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS

"Contribution" means:

a) in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and b) in the case of each subsequent Contributor:

i) changes to the Program, and

ii) additions to the Program;

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.

"Licensed Patents " mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.

2. GRANT OF RIGHTS

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.

b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

a) it complies with the terms and conditions of this Agreement; and

b) its license agreement:

i) effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;

ii) effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;

iii) states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and

iv) states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

a) it must be made available under this Agreement; and

b) a copy of this Agreement must be included with each copy of the Program.

Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so

in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further

action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against a Contributor with respect to a patent applicable to software (including a cross-claim or counterclaim in a lawsuit), then any patent licenses granted by that Contributor to such Recipient under this Agreement shall terminate as of the date such litigation is filed. In addition, if Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. IBM is the initial Agreement Steward. IBM may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

Special exception for LZMA compression module

Igor Pavlov and Amir Szekely, the authors of the LZMA compression module for NSIS, expressly permit you to statically or dynamically link your code (or bind by name) to the files from the LZMA compression module for NSIS without subjecting your linked code to the terms of the Common Public license version 1.0. Any modifications or additions to files from the LZMA compression module for NSIS, however, are subject to the terms of the Common Public License version 1.0.