

CA PPM v14.3

Studio Projects  
Release 1.9 – User Instructions

# Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Author | Date | Version | Comments |
| Alexandre Assis | 05/Feb/2009 | 1.0 | * Initial Draft |
| Alexandre Assis | 10/Feb/2009 | 1.1 | * Known Issues * Application Menu Packaging * Reports and Jobs Description Packaging * Support for easier Globalization (Translations) * Links in the Element List |
| Alexandre Assis | 03/Mar/2009 | 1.2 | * New Features: discovery, packaging and translation of Stock Object Custom Attributes; exporting the StudioProject itself * Bugs fixed in Discovery and Packaging |
| Alexandre Assis | 16/Mar/2009 | 1.3 | * New Features: two types of Data packaging: seed data (to be included in installation) and demo data (has its own optional installation); publish file: a ZIP file will be created under “$CLARITY\_HOME\webroot\StudioPackages\<ProjectCode>” and a link is created in the Studio Project record, so one can download the zip file without having to access the file server; if packaging folder is left blank, a temp folder will be automatically created as “$CLARITY\_HOME\webroot\StudioProjects-Temp\<ProjectCode>” ; new object “Clarity Instance” allows for the re-use of servers without the need to type all of the information again. Instead of pointing to a XOG URL, User, Password, etc, you say the “Targed Instance”. * Lots of bugs corrected in Translation, Packaging and Discovery * Enhancement: Performance and memory usage improvement: XOG Login and XOG Logout functions have been segregated in their own specific Scripts – XOGURL, XOGLogfolder and sessionID are persisted in XOG Login so all of the scripts use the same session, eliminating the need to login and logout over and over again – and facilitating StudioProjects Installation procedure as only one set of parameters exist now; Now the PrepareEnvironment script will create the temp folder by itself. GEL can actually create folders!!!; Installation file will now consider the SSLENABLED var; |
| Alexandre Assis | 21/Jul/2010 | 1.6 | * Several bugs fixed in Discovery, Packaging and Translations * New feature: Studio Projects now handles Stock Objects better * New feature: Studio Projects now handles three levels of objects: Master, Subobject and SubSubObject * SP5 introduced “Actions” tag that are now considered |
| Alexandre Assis | 20/Dec/2010 | 1.6 | * Separated Installation procedures in a different document |
| Alexandre Assis | 13/May/2011 | 1.6.1 | * Bugs fixed * Tested for v12.1 * Updated User Guide with Complete list of Element Types |
| Alexandre Assis | 23/nov/2011 | 1.7 | * Tested for v13 Beta * Re-designed “Element Link“ for v13 |
| Alexandre Assis | 14/jul/2014 | 1.7.4 | * Added information on v13.3 xog changes |
| Alexandre Assis | 14/01/2015 | 1.7.7 | * Changes for Packaging and Publishing |
| Alexandre Assis | 10/dec/2015 | 1.8 | * Added OBS Packaging * Added UITheme Packaging * Objects file broke in two – with and without actions – for the appropriate installation moment (pre-processes, post-processes) * New option to package ONLY the content that is additional to CA PPM Content Manager |
| Alexandre Assis | 15/apr/2015 | 1.9 | * Added XOG Governor for Data packaging * Added Port number to XOG SOAP Calls to avoid Internet Routing on Linux * Added stproj\_file object to store file names * Re-designed logs to web folder for debugging * Re-designed installation files using stproj\_file data * StockObjectAttributes broken by attribute for the inv object |

**Table of Contents**

[1. Version History 2](#_Toc437545445)

[2. Studio Projects – Using Studio Projects 5](#_Toc437545446)

[2.1. Create a new Clarity Instance 6](#_Toc437545447)

[2.2. Create a new Studio Project 7](#_Toc437545448)

[2.3. Running the Discovery Process 10](#_Toc437545449)

[2.4. Translating Elements 13](#_Toc437545450)

[2.5. Packaging Data 16](#_Toc437545451)

[2.6. Packaging your Studio Project 17](#_Toc437545452)

[2.7. Creating your final Package 19](#_Toc437545453)

[3. Future Steps 20](#_Toc437545454)

[4. Curious fact 21](#_Toc437545455)

# Studio Projects – Using Studio Projects

This functionality is not to be demonstrated, but to be used by Technical Sales and Services in packaging stuff they build for prospects and customers for later reuse.

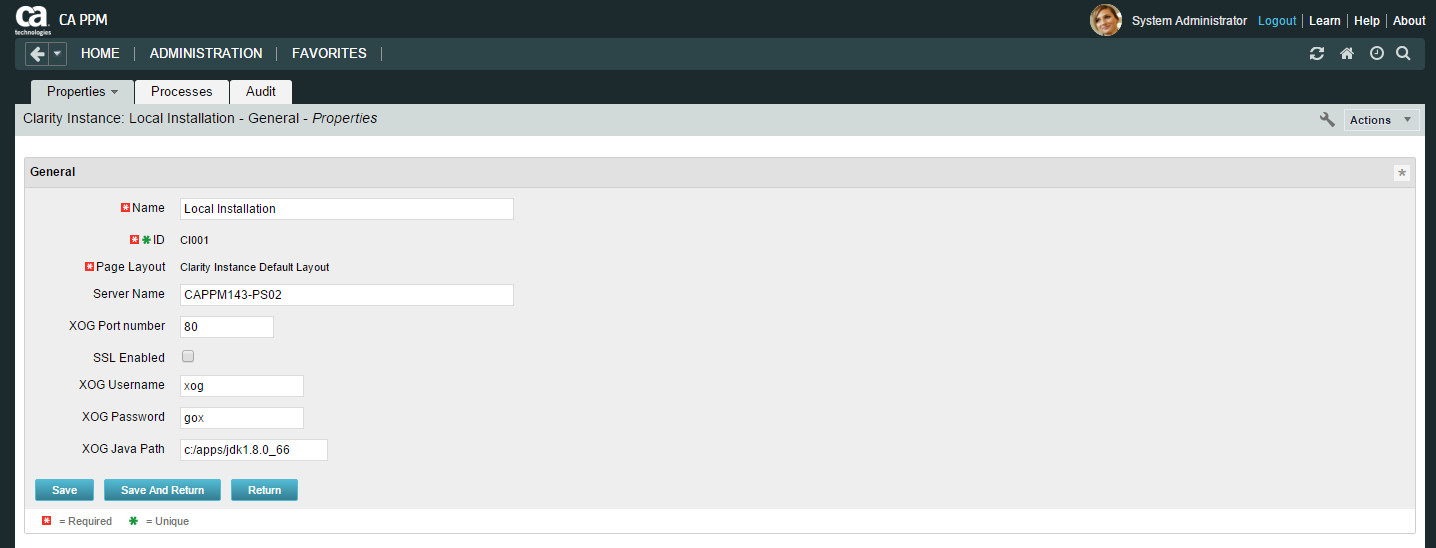
This is basically what it does:

1. You create a new Studio Project
2. You run a “Discovery Process”. This process will try to automatically find the Studio Elements of your project based on information you provide in the attribute “Project Initials”. Studio Elements in this scope are defined as: Static Lookups, Dynamic Lookups, Master Objects, Sub Objects, Stock Objects, Processes, Queries, Portlets, Pages, Groups, Reports, Jobs, OBS, UI Themes and the Application Menu.
3. You also define Data you need to package as well as studio elements. You can package investments, custom object instances, and many other objects. “Seed Data” refers to data that is required for the system to operate correctly and is included in the Installation batch. “Demo Data” refers to optional test data that may or may not be necessary depending on the situation.
4. You adjust the elements as needed and run the “Packaging Process”.
5. The Packaging process will create all XML XOG files and an Installation batch file for your package. The Installation Batch file is prepared to run all XOG files in the correct order for you to be able to install your new package in another Clarity installation.
6. Your package is 99% release-independent – because we don’t include much Stock Object information in it. Exception is when there is a major change in Clarity Studio tables. Example: Packages created pre-v12.0.3 need to be manually edited to work properly due to a change in how Clarity Studio handles the List View.

Keep reading for the details on how to use.

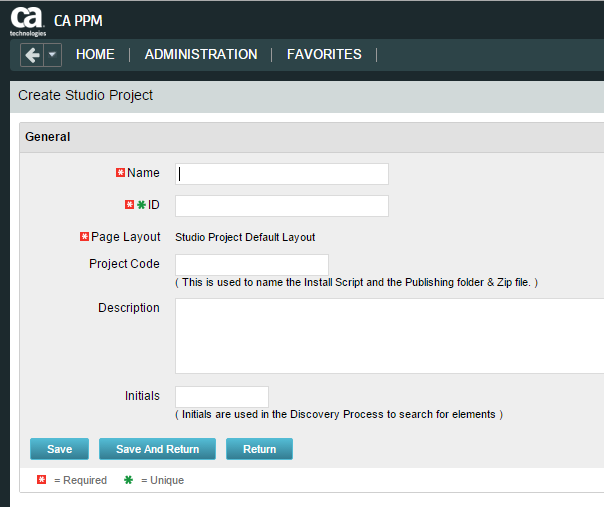
## Create a new Clarity Instance

Clarity Instances are used to hold server-dependent parameters. This package already creates one Instances for you called Local Installation. You must adjust the server name appropriately. Instance data is used to create the Installation Batch file.

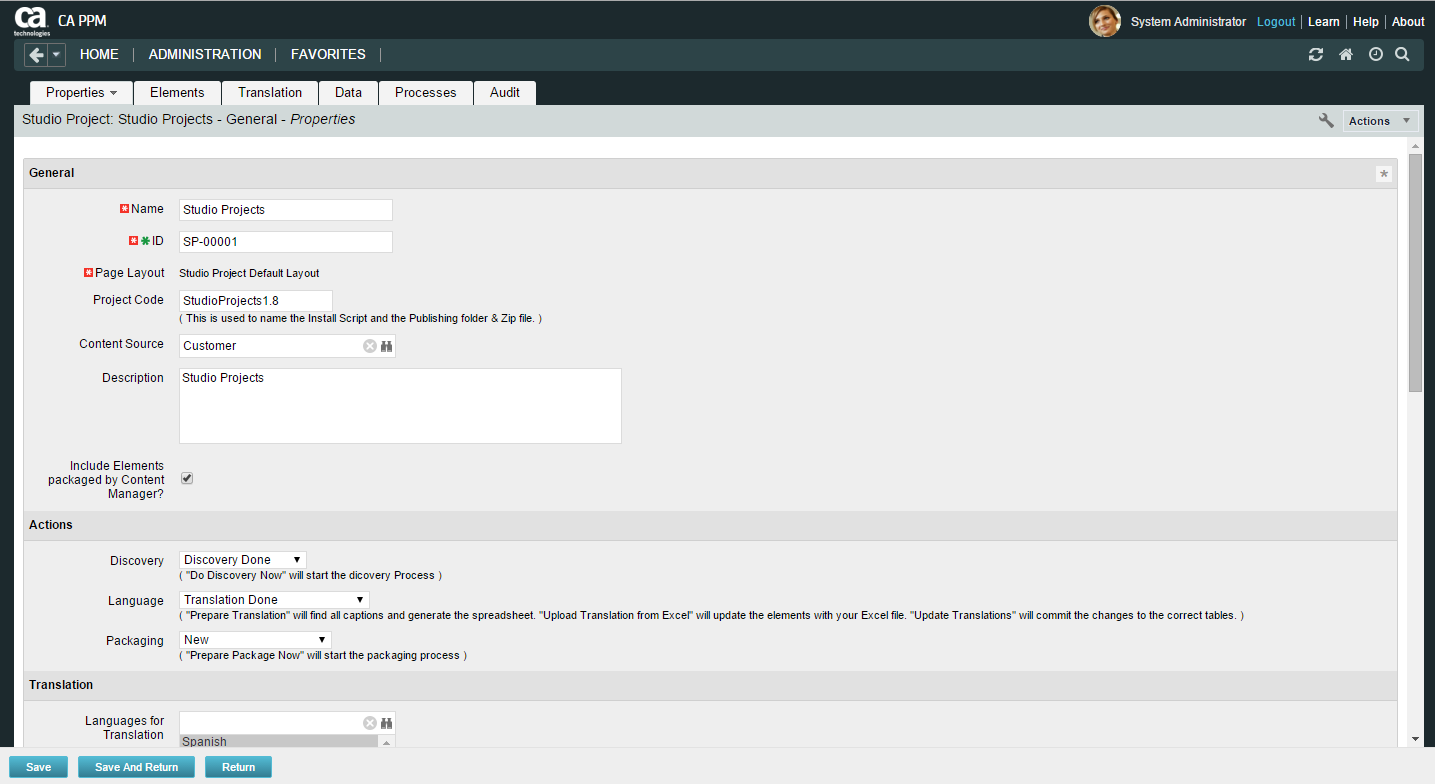


## Create a new Studio Project

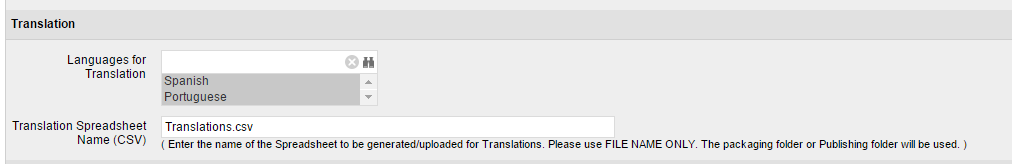
1. Navigate to Section “Studio Packaging”, Menu Item “Studio Projects”
2. Click New and create a new Studio Project
3. Enter the information required. Please notice the Hints guide you on using each attribute.



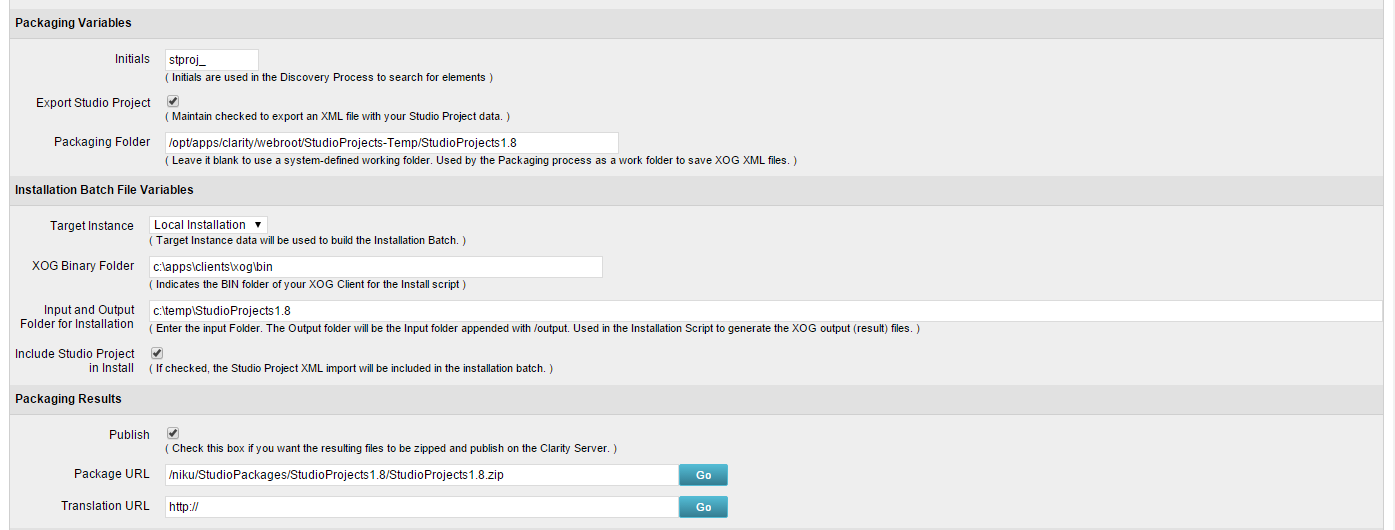
1. Save. You will see more information. The actions section contains the three different types of Actions you can start from within your project: Discovery, Packaging and Translations.
   1. The Discovery field is used to trigger the Discovery Process. We’ll talk about it later.
   2. The Language field is used to trigger the Translation Processes. We’ll talk about it later.
   3. The Packaging field is used to trigger the Packaging Process. We’ll talk about it later.



1. Fill in The Translation section if necessary. This functionality is optional.
   1. Choose the languages you will translate your application into
   2. Enter the filename ONLY of the CSV file to be generated.



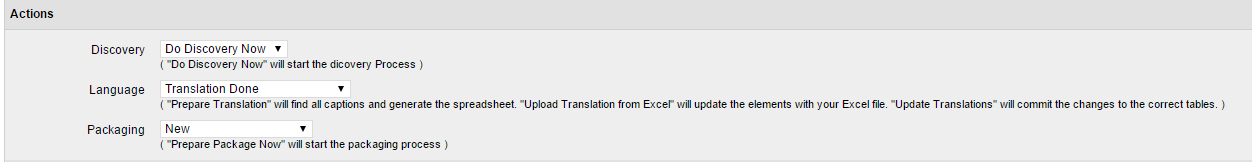
1. Fill in the Packaging Variables. Notice the Hints will guide you in using those variables.
   1. Initials: if you used initials to identify your elements when you’ve built them, Studio Project’s Discovery Process will be able to find them for you automatically.
   2. Packaging Folder: this is the place used by the Packaging process to save your package’s files. The translation spreadsheet will be included in the same folder UNLESS you use the “Publish” option below.
   3. Export Studio Project: Check to enable the generation of an XML file containing this StudioProject’s data for upload in a target Clarity instance if necessary or just to save a list of the elements.
2. Fill in the Installation Batch File Variables. Notice the Hints will guide you in using those variables.
   1. Target Instance and XOG Binary Folder are used to indicate to your Installation Script how to run XOG.
   2. Installation Input and Output Folder will be used to determine input and output folders for your Installation Script, to store xog results when you run it.
   3. Include Studio Project in Install: Check to create an entry in the installation batch file for uploading the StudioProject’s data into the target instance. The target instance MUST have StudioProjects functionality installed for this option to work correctly. Otherwise, do not check this option or comment out the corresponding line in the batch file.
3. On Packaging Results, check “Publish” if you want to create a ZIP file under Webroot and store a Package URL (automatically created) for linking/downloading purposes. The Translations spreadsheet will be created in the same location and there will be a link available for each one.



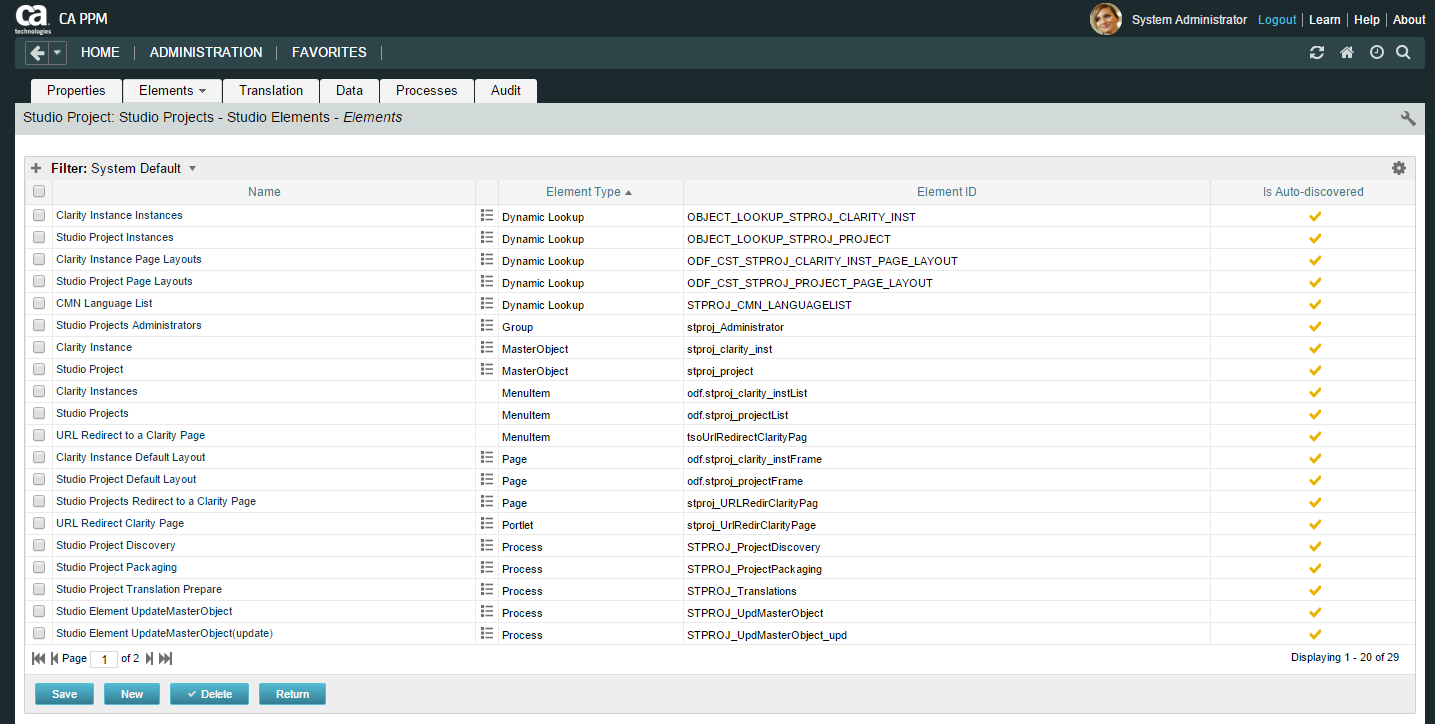
## Running the Discovery Process

The Discovery Process is designed to automatically find project elements based on the defined project initials. For instance, the “Studio Projects” functionality has “ST\_” as its initials. Therefore, it will search the database for Objects, Queries, Portlets, Pages, Lookups, Groups, MenuItems and Processes that start with “ST\_”.

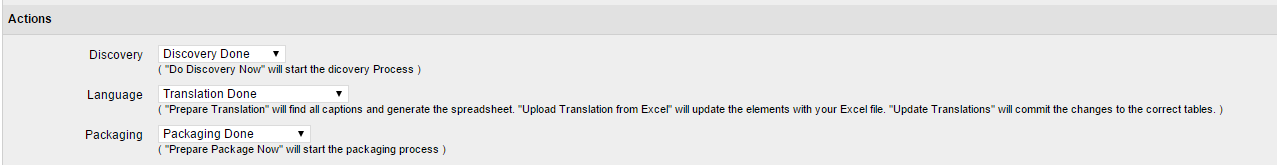
1. Change your project’s “Discovery” action to “Do Discovery Now” and Save.



1. Monitor the Process success. After it finishes, Navigate to “Studio Elements”. Your Element List should show something like the picture below. Please note not all Projects contain all elements, so your list may be very different from my example:



1. Notice there is an icon that will take you directly to that specific item if needed.
2. Check your Element List. If Elements are missing, click “New” and add them manually. If Elements unrelated to your project were found, delete them.
   1. NOTE1: the effectiveness of the Discovery process is 100% dependent on your ability to use a set of Initials to identify your elements when you create them. Preferably, it should be something unique.
   2. NOTE2: explanation on MasterObjects and SubObjects:
      1. MasterObjects ALWAYS carry their own SubObjects, so you don’t need to add SubObjects when the MasterObject is already there.
      2. Therefore, the SubObjects Element Type should only be used when you want to export ONLY the SubObject, without taking the whole MasterObject. This is very useful for exporting SubObjects of Clarity Stock Objects, like Investments or Projects, for instance. To keep your package as close as possible to be release-independent, the Stock Objects should not be exported – not to interfere with the destination system. An Example: Suppose you create “Request” as a subobject of “Project”. Adding it as a SubObject to your Studio Project will carry only the “Request” object and not the whole “Project” object into your package. This ensures your target system’s Project object will not be overwritten.
      3. The Discovery Process is smart enough to realize when to include a SubObject or not. Just use your Initials and the process will do the work for you.
   3. NOTE3: during MasterObjects discovery, its related dynamic lookups, pages and menu items are also discovered
   4. NOTE 4: during Pages discovery, its related menu items are also discovered
   5. NOTE 5: if you include a MasterObject manually and run the Discovery Process, its related dynamic lookups, pages and menu items will be discovered
   6. NOTE 6: if you include a Page manually and run the Discovery Process, its related menu items will be discovered
3. “Is Auto-discovered” marks all items that have been included by this process. You can run the Discovery process as many times as necessary. The items you add manually WILL NOT BE DELETED by the Discovery process. Therefore, always check for duplicates!!
4. Navigate back to your General Subpage. Check the “Discovery” status: it should say “Discovery Done”.



1. Here’s a complete list of the available Element Types:

|  |  |
| --- | --- |
| Studio Element Type | How does it work? |
| Static Lookup | Used to export a Static Lookup. |
| Dynamic Lookup | Used to export a Dynamic Lookup. |
| MasterObject | Used to export Custom Objects. The Object and all of its children will be exported. Dynamic Lookups, Pages, Menu Items will ALSO be exported and DO NOT need to be explicitly declared. |
| SubObject | Used to export Custom Sub Objects, when you don’t want to export the Master Object; it is especially useful when the Master is a Clarity Stock Object such as the Project Object and you don’t want to export it. |
| Stock Object | Used to export all custom attributes of a Stock Object that match the Project Initials, as well as the object’s Views |
| Stock Subobject | Used to export all custom attributes of a Stock Sub Object (e.g. the Risk or the Issue objects) that match the Project Initials, as well as the object’s Views. |
| Stock Object Custom Attribute | Used to export a Stock Object Custom Attribute. The resulting XML file will contain ONLY the Custom Attribute and NOT the complete list of attributes of the Clarity Stock Object. |
| Stock Object List View | Used to export just the List & Filter Views of a Stock Object. |
| Stock Object Properties View | Used to export just the Create and Edit Views of a Stock Object. |
| Process | Used to export a Process. |
| Query | Used to export a Query. Database elements such as Stored Procedures and Functions are NOT exported. |
| Portlet | Used to export a Portlet. |
| Page | Used to export a Portlet Page. |
| Group | Used to export a Group. |
| MenuItem | Used to export Menu Items. The Menu will not be completely exported. Only the declared Menu Item will be exported. This may require adjustments after the Import is done, depending on how the Application Menu was in the target system. |
| Report | Used to export a Report Definition. The Report executable (Crystal, WEB Intelligence) is NOT exported. |
| Job | Used to export a Job Definition. The Job code (Stored Procedure, Java Class, etc) is NOT exported. |
| OBS | Used to export an OBS structure (useful or security) |
| UI Themes | Used to export UI Themes |

## Translating Elements

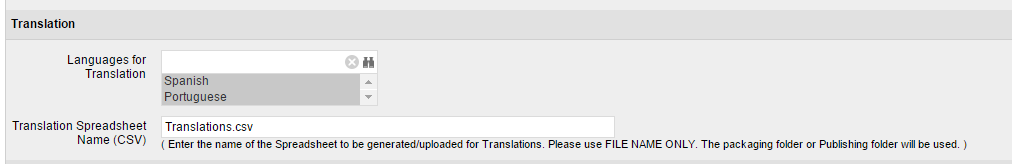
Translating Stuff is always a pain. Too many clicks, saves, submits… a few bugs that make you lose your translation work, etc.

Therefore, “Studio Projects” includes an easier way of translating studio elements. It will search the database for all element’s labels and create “Studio Element Translations” in your Project.

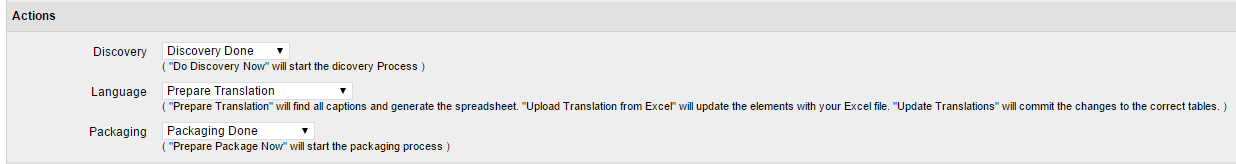
You can: use edit mode to enter translations directly into Clarity OR use a CSV file generated by the process to do your translations in EXCEL.

After that you can: upload from Excel or just update the original labels with the new translated ones.

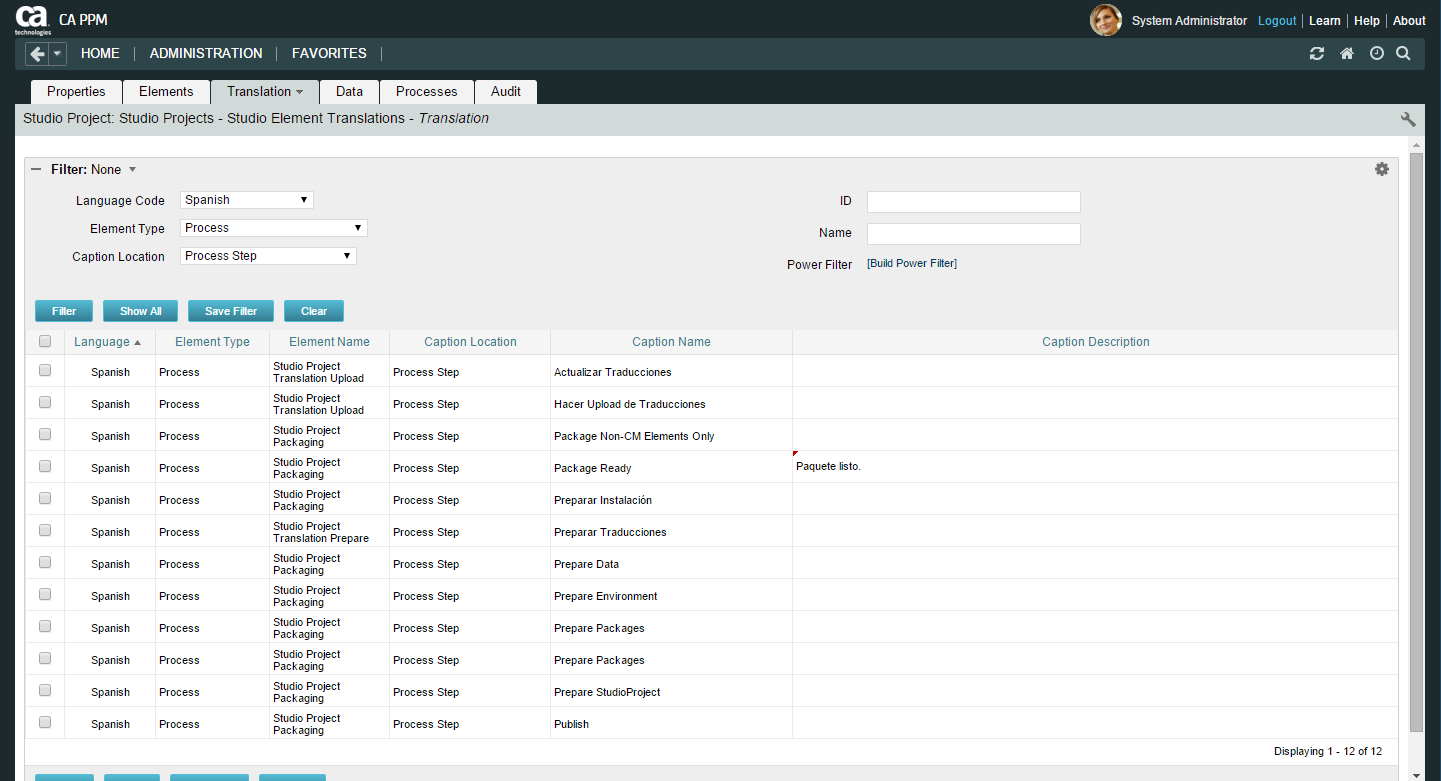
1. First step is to “Prepare translations”. Verify that all your target languages are set and your output file is defined. Even if you are not going to use it, the file is created anyway so enter a valid path and filename. In my example, I’m translating into Portuguese and Spanish only.



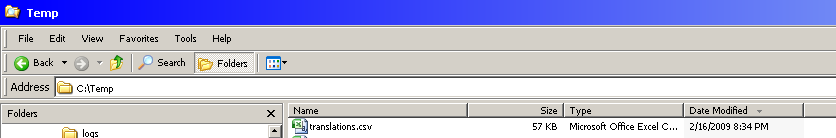
1. In the Actions Section, change Language to “Prepare Translations” and Save. This will trigger a Process that will prepare your application to be translated.



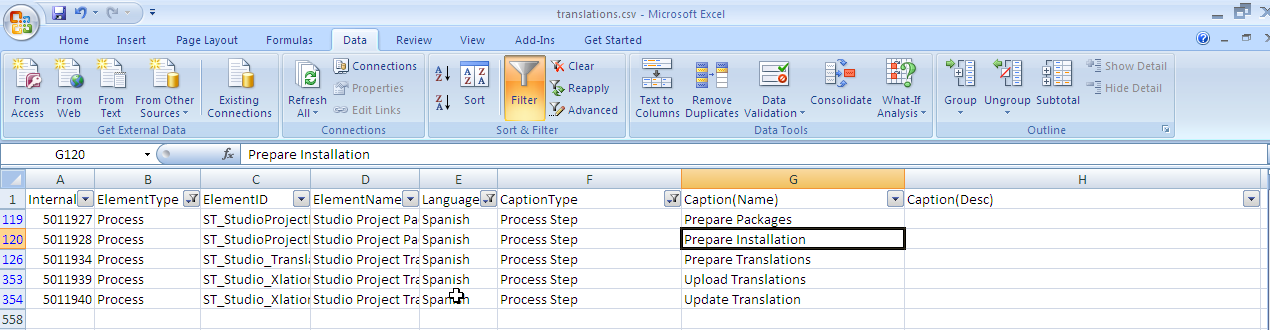
1. When the process is finished you the action will be automatically set to “Waiting for Translation”
2. Now, it’s your turn: navigate to “Translations”. In the example below, we are filtering “Spanish”, “Process”, “Process Step”. These are the labels to be translated. Both “Name” and “Description” fields are available. You can enter Edit Mode and translate everything directly in Clarity pages, but this is not what we will do now.

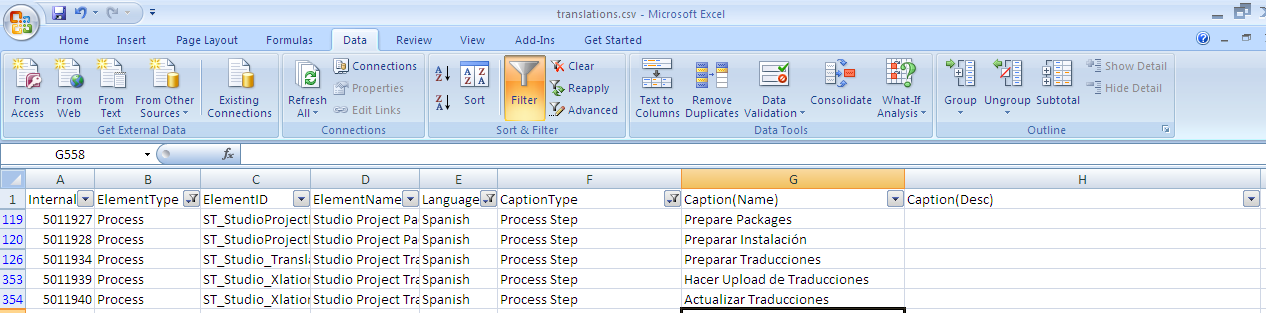


1. Navigate in Windows Explorer to the path you specified for your translation CSV file. Open that file in Excel.



1. You can use “Data/Filter” to set up an autofilter in Excel to make your translation effort easier. Translate all you need and save.





1. Now, back to Clarity, in “Actions”, you must select “Upload Translation from Excel” and save.
2. When the process has finished, you will notice the action changes to “Ready to Update”. This will allow you to check the translations before committing them to the database.
3. You can navigate to “Studio Element Translations” and you can see the translations have been uploaded from Excel.

How does it work internally?

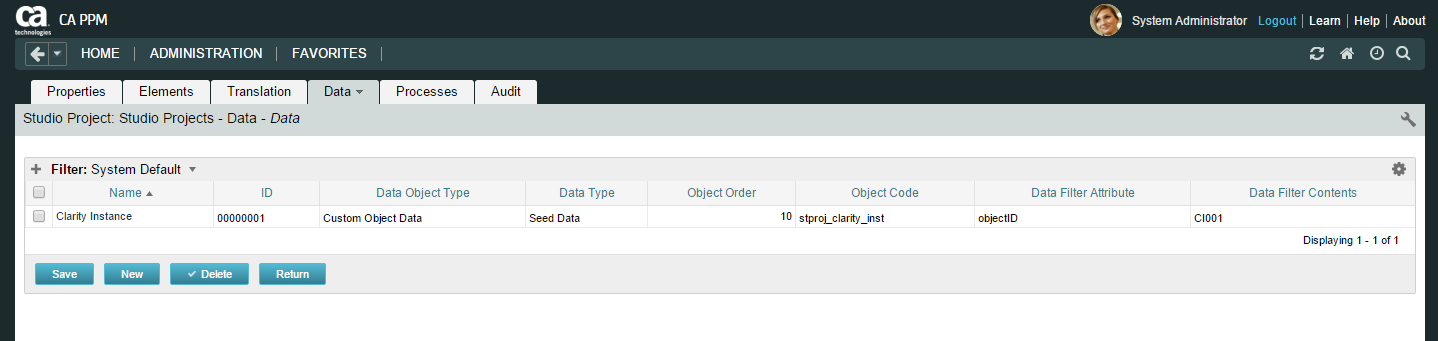
Prepare Translations uses XOG to create new “Studio Element Translation” instances. It searches the database using the Element List and creates a reference to all Captions that need translation. The Caption’s table **(CMN\_CAPTIONS\_NLS)** internal id is kept as reference for later update.

Upload from Excel will update Studio Element Translations and then continue to step “Update Translations”, which directly updates the CMN\_CAPTIONS\_NLS table with the new captions using its internal id as reference.

1. When you’re ready to do so, change the Language action to “Update Translations”. This will commit the values in the Element Translation list into Clarity’s Captions table, as well as clear your StudioProject’s Element Translation list.
2. The final status is “Translations Done”.

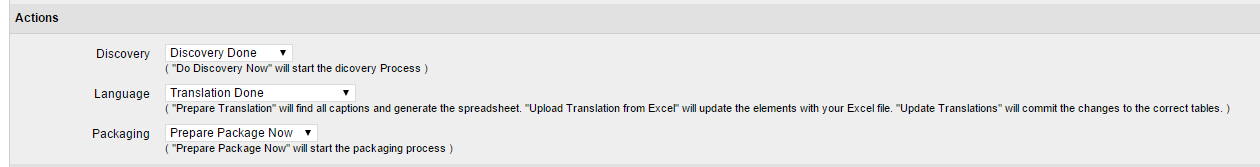
## Packaging Data

1. You may want to package data for two reasons:
   1. It’s part of your application setup, therefore needed for your application to work. This is referred to as “Seed Data”.
   2. It’s demo data used for demo purposes. This is referred to as “Demo Data”
2. Navigate to “Data” and include the data elements you want to package.
   1. Seed data will be included in the Installation Script.
   2. Demo data will have its own separate Installation Script, therefore being optionally installed or disregarded when necessary.
3. Elements can be Stock Objects or Custom Objects
   1. For Stock Objects, enter the Data Field Attribute – field to be used for filtering - as well as the Data Field Contents - values of that field you want to consider; Example: To export a project with ID “PR1000”, you must include Data Field Attribute = “projectID” and Data Field Contents as “PR1000”. Notice contents can contain multiple values separated with a comma, such as “PR1000, PR1001, PR1002”. The Object Code is ignored for Stock Objects, as it’s pre-defined by the Data Object Type you choose.
   2. For Custom Objects, enter the Object Code – code of the custom object – as well as the Data Field Contents – IDs of the instances you want to export; Example: To export a “Studio Project with ID “SP-00001”, you must include Object Code “studio\_project” and Data Field Contents as “SP-00001”. Just as vefore, contents can contain multiple values separated by commas. The Data Field Attribute is ignored for Custom Objects, as it’s always “instanceCode”.

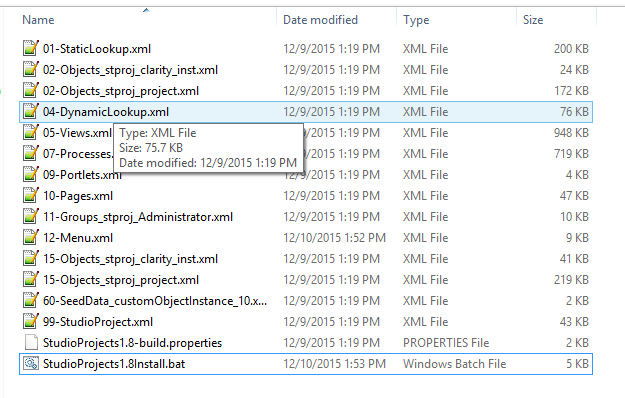


## Packaging your Studio Project

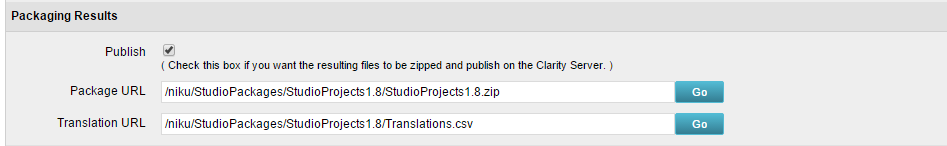
1. Create your package folder, as described by the “Packaging Folder” attribute of your project
2. Change your project’s “Packaging” action to “Prepare Package Now” and Save.



1. Monitor the Process success. After it finishes, open your Packaging Folder and verify the contents:
2. If you have access to your Packaging folder you can see it, this is what it should look like:



1. If you use the “Publish” feature, this is where your published file will be in <CLARITY\_HOME>/webroot/StudioPackages
2. But you don’t need access to0 the server because a URL will be provided for you so you can easily download it:



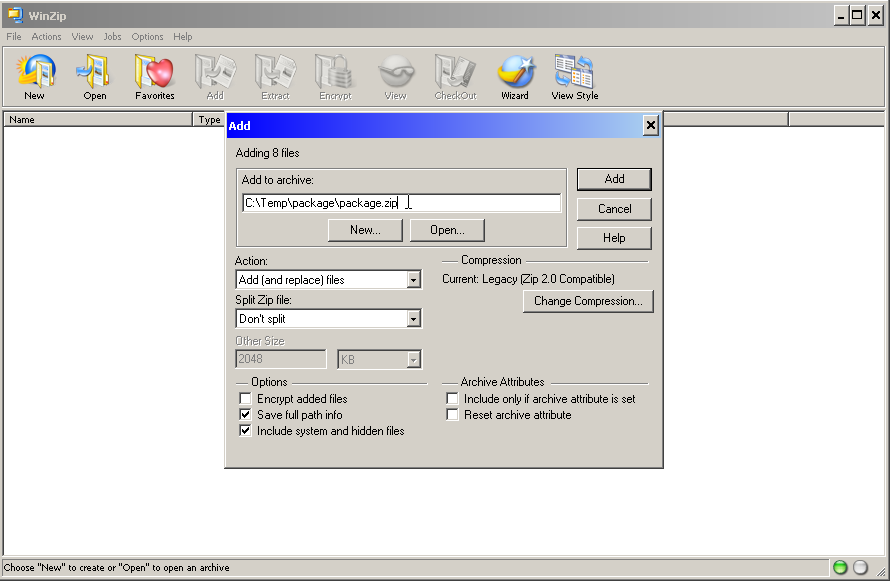
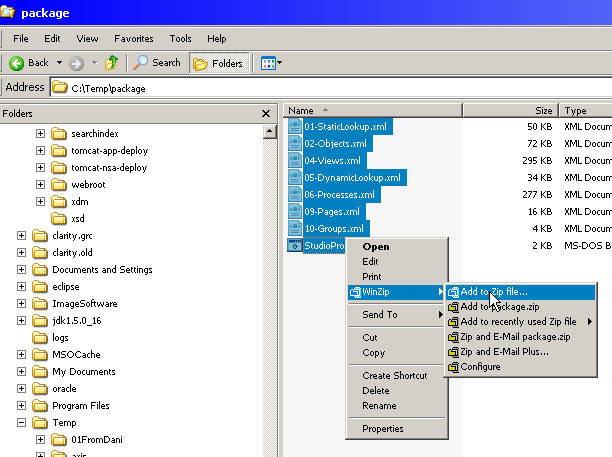
## Creating your final Package

If you chose to publish your package, you can download it using the link in the Studio Projects List.

This will be a ZIP file stored under $CLARITY\_HOME\webroot\StudioPackages\<ProjectCode>.

To create your own package using the packaging folder, follow these steps:

1. Navigate to your package’s folder and select all files. Right Click and choose “Winzip\Add to Zip file…”



1. Enter the Zip Filename and make sure you have “Save full path info” checked.
2. Click Add. Your Zip file is ready to be shipped!

# Future Steps

1. Studio Projects is in Maintenance Mode due to the release of the “Content Management” functionality which is basically the same.
2. Studio Projects will be eliminated once Content Management gets to the same level.

# Curious fact

The “Studio Projects” application was package using “Studio Projects”. ☺ It actually packaged itself!