

Publishing GRIN-Global Accession Data with the Genesys Wizard

Contents

Background	1
Installing the Genesys Wizard.....	1
Application Configuration	2
How to Publish Data.....	2
Setting Up the Genesys Account.....	2
Configuring the Connection	2
Uploading Data	4
Deleting Data	5
Implementation Approach.....	6
Genesys .Net API Library.....	6
Core Classes	6
Securely Storing Credentials	7
Building a Custom Dataview	7

Background

Blue Bicycle Technologies was contracted by CIMMYT to migrate the existing GRIN-Global Curator tool integration with Genesys to a separate Curator Tool Wizard. The migration will allow the Genesys Wizard to be distributed without the need to modify the core Curator Tool. This document provides instructions on how to use the new functionality, as well as a brief overview of the design approach.

Installing the Genesys Wizard

The Genesys Wizard functionality is built leveraging the standard pluggable Curator Tool wizard framework. The wizard is compiled into a DLL called GenesysWizard.dll, which is placed within the *Wizards* subdirectory under the Curator Tool executable.

The GenesysWizard.dll relies on two supporting DLLs which must be placed in the same directory as the Curator Tool executable. This is necessary so that supporting DLLs can be found when the Wizard is loaded. Those two DLLs are:

- GeneSys2.Client.dll
- Newtonsoft.Json.dll

Application Configuration

By default, the Genesys Wizard expects to pull passport data for transfer from a known dataview called "get_passport_data". The dataview can be overwritten by adding a Curator Tool Application Setting with the name of "GenesysWizard-DataView".

How to Publish Data

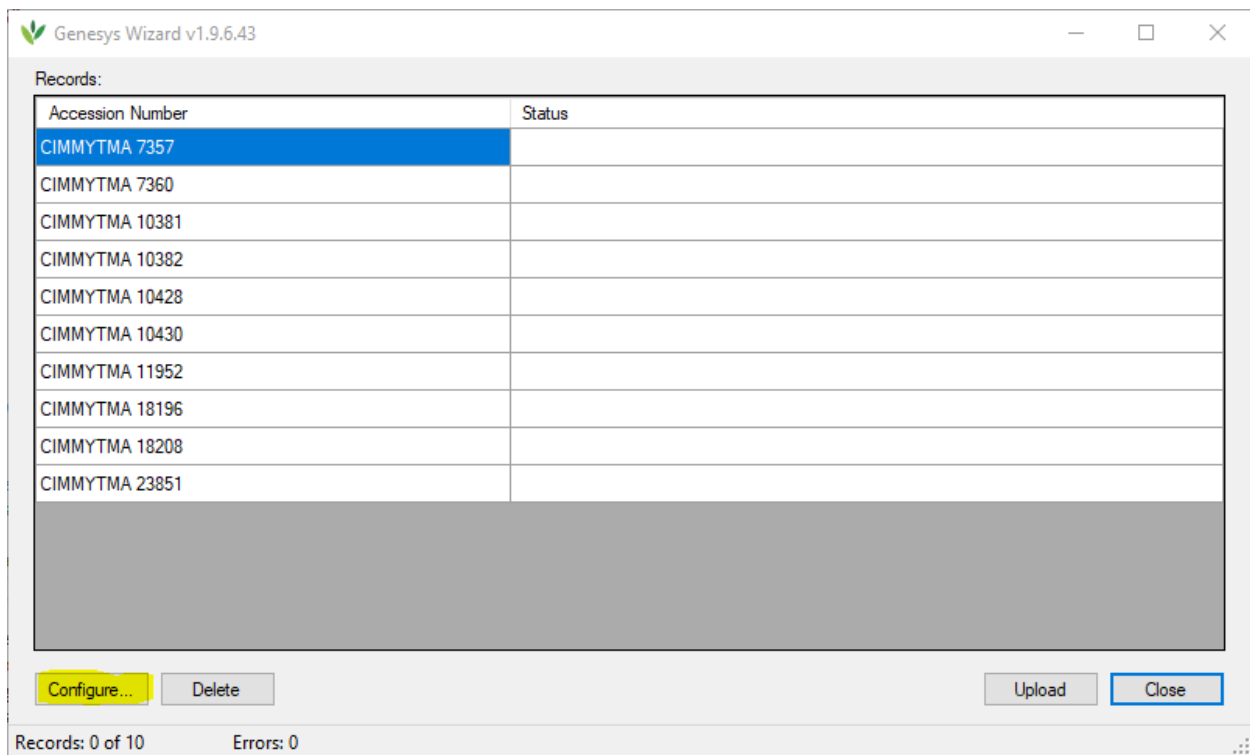
Before accession data can be sent, the user must create an account on Genesys and configure the connection to authorize the Curator Tool to send data on the user's behalf.

Setting Up the Genesys Account

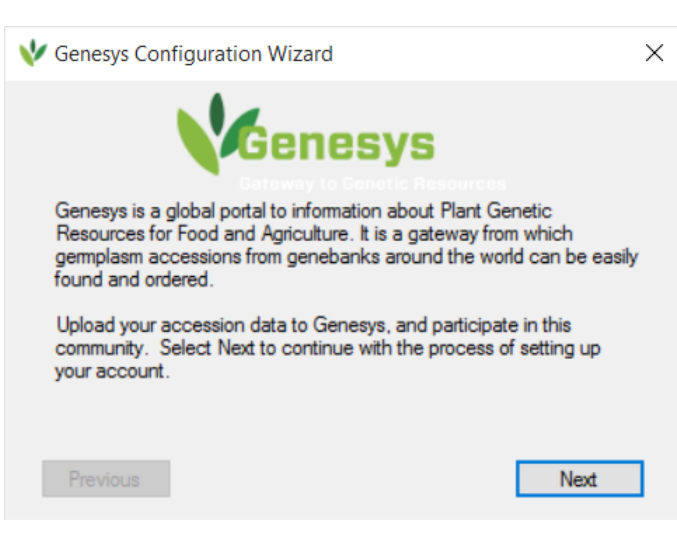
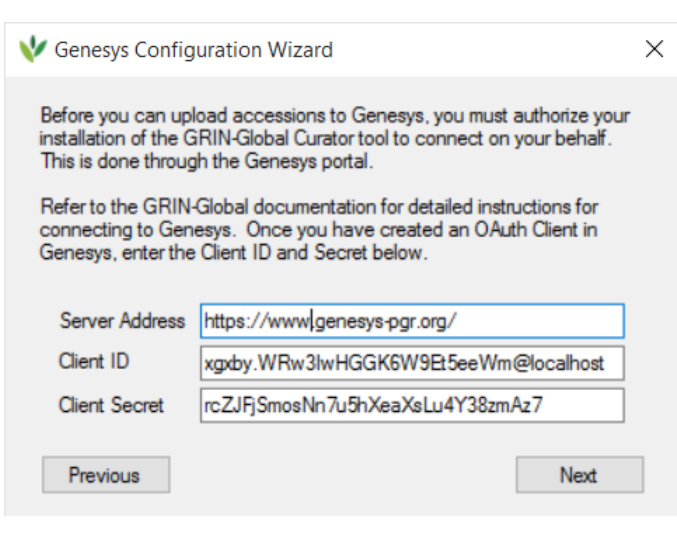
Visit the Genesys site at <https://www.genesys-pgr.org> and register. Once you have registered, request access to publish data by sending an email to helpdesk@genesys-pgr.org. You will be provided with two text values called the Client ID and Client Secret, which will be used to configure the connection from the Curator Tool to Genesys.

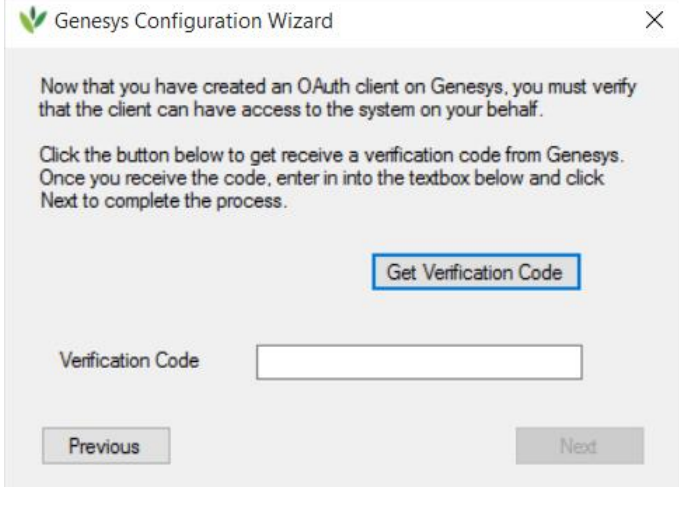
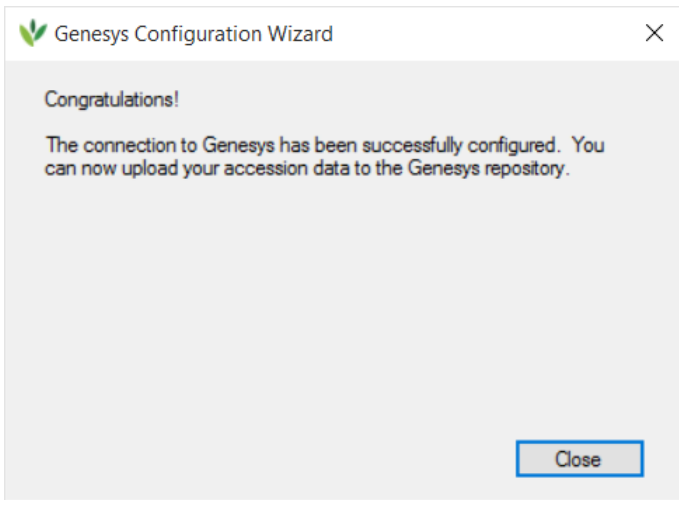
Configuring the Connection

To setup the connection to Genesys, a configuration tool was implemented to walk the user through the process. To access the tool, launch the Genesys Wizard from the toolbar and click the "Configure..." button.



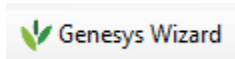
The wizard will take you through the following steps:

	<p>Welcome screen introduces the Genesys product.</p>
	<p>The first step is to enter the Client ID and Client Secret provided by the Genesys website when you setup an account.</p> <p>The Genesys server address will default to the production environment located at https://www.genesys-pgr.org/. It can be changed to another destination if needed for testing.</p>
	<p>Once the OAuth client has been created and entered in the prior screen, you must authorize the Curator Tool to act on your behalf.</p> <p>Click the “Get Verification Code” button to launch a browser window, and login to the Genesys site.</p> <p>Agree to allow access for the curator tool by clicking the “Yes, allow access” button. The resulting page will provide and authorization code. Copy this value into</p>

	<p>the “Verification Code” textbox and select Next.</p>
	<p>If the process was successful, the dialog to the left will be displayed. If there was a problem, the user will be notified of the error and can modify the information and resubmit.</p>

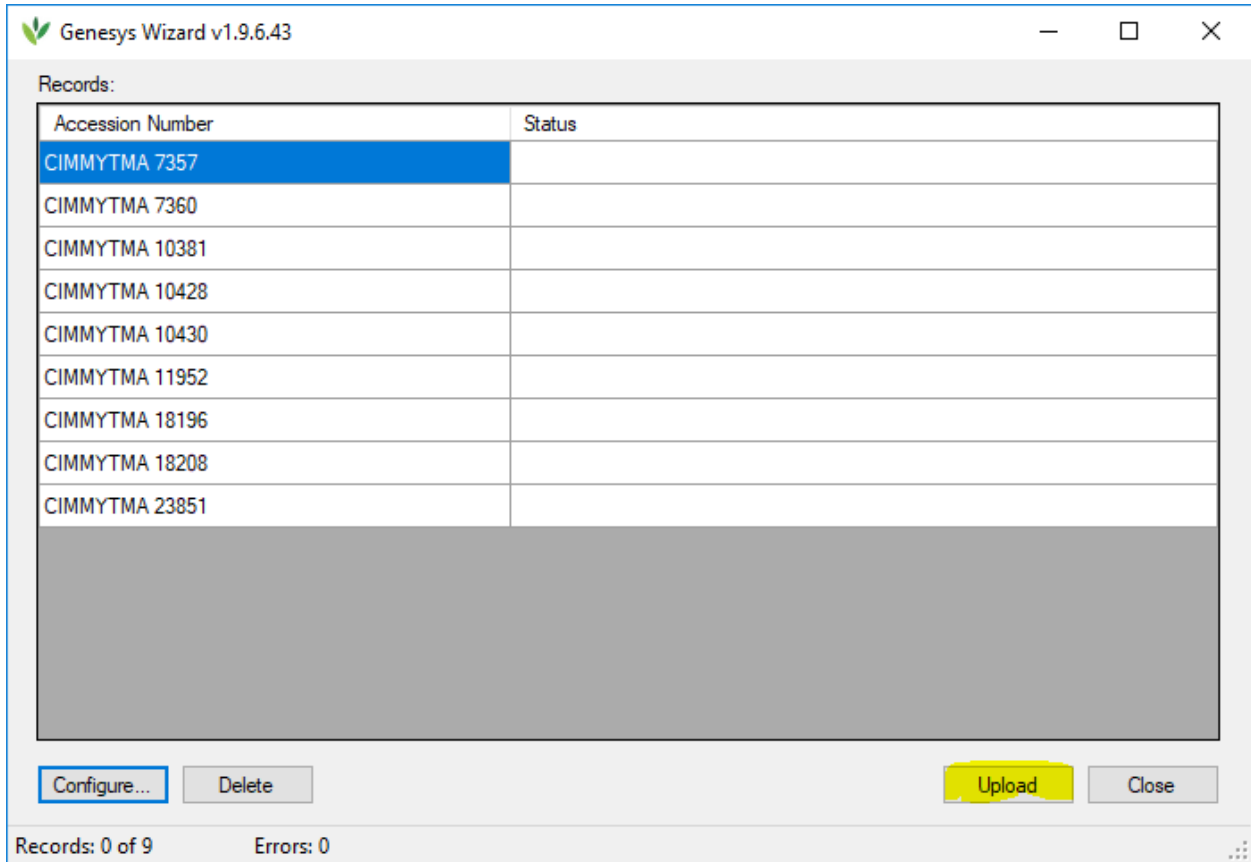
Uploading Data

To publish accession information to Genesys, start with an accession dataview within the Curator Tool and highlight the rows you want to transfer. Once highlighted, select the Genesys Wizard from the toolbar.



When the Genesys Wizard is launched, it will interrogate the passport dataview to verify that the required data elements exist. If data elements are missing, a dialog box will display which lists the missing pieces of information. Once the dataview is corrected, the Genesys Wizard will function.

The accession records in the list are the records that were highlighted for transfer. Once the “Upload” button is selected, accession records will be sent to Genesys in batches and the individual status of each accession will be updated on the screen.



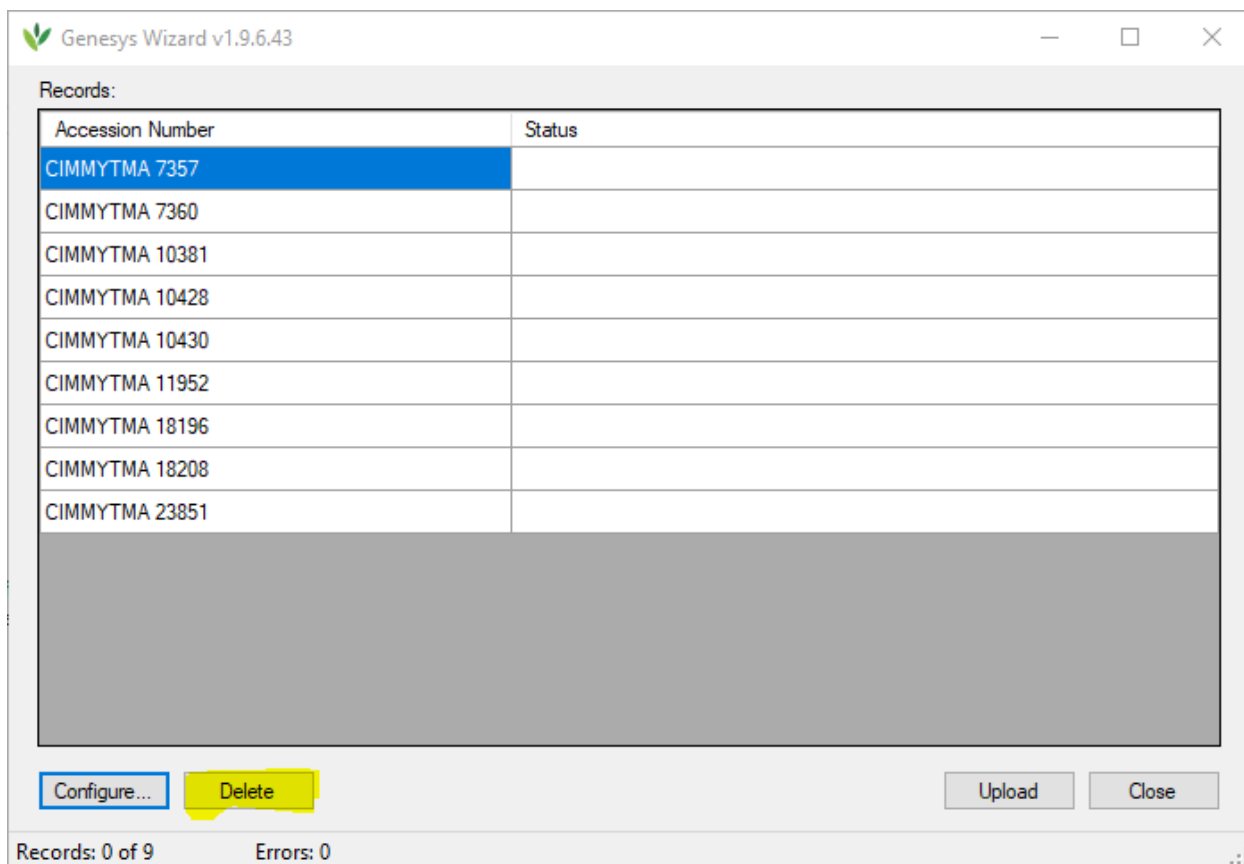
The upload process can be run multiple times without any negative effect.

Note: If the Upload button is not enabled, then there are connectivity issues with Genesys (for example the authentication has expired). Rerun the configuration tool to authenticate the application.

Deleting Data

Similar to publishing accession information to Genesys, the process of deleting accession information from Genesys is started by selecting one or more accession records and launching the Genesys Wizard.

Selecting the “Delete” button rather than the “Upload” button will mark the accession records in Genesys for deletion.



The delete process can be run multiple times without any negative effect.

Note: If the Delete button is not enabled, then there are connectivity issues with Genesys (for example the authentication has expired). Rerun the configuration tool to authenticate the application.

Implementation Approach

Genesys .Net API Library

The integration with Genesys is coordinated by the Genesys .Net API library, created by Blue Bicycle Technologies for The CropTrust. The library simplifies the Genesys authentication scheme, constructing well-formed requests, and parsing responses. Further, the library provides a number of strongly-typed classes representing Genesys object. In order the leverage the Genesys .Net library, the GRIN-Global projects were upgraded from .Net 3.5 to .Net 4.5.2.

Core Classes

The vast majority of the code to interact with the Genesys .Net API library is centralized in the Genesys directory of the GRINGlobal.Client.Common project. The core classes are as follows:

Class	Purpose
SyncController.cs / ISyncController.cs	Central point of entry for all Genesys operations. The class exposes the following key methods: CheckConnectivity() – Verify connectivity with Genesys

	Configure() – Launch configuration wizard dialog UploadAccessionData() – Launch upload data dialog DeleteAccessionData() – Launch delete data dialog
SecureTokenStorage.cs	Stores user Genesys authentication information
GenesysResources.resx	Resource file containing user messages and strings. Allows them to be localized for different languages.
ConfigWizard.cs	Configuration wizard dialog and associated logic
GenesysWizard.cs	Upload / delete access dialog and associated logic

Securely Storing Credentials

Genesys credentials are per individual user, so the implementation was designed to ensure that the credentials are stored securely. It handles the case where each user logs into GRIN-Global from a unique Windows account, or when they login with their individual logins from a shared Windows account.

Credentials are stored in a text file located in the Window user's application data folder. Specifically, it is stored at C:\Users*<username>*\AppData\Roaming\GRIN-Global\Curator Tool. A file is created for each curator username. For example, if a user with username *joeblow* configures a Genesys connection, a file names *genesys_joeblow.txt* will be created containing client id, client secret, and tokens.

Only users who login into the machine with the same Windows user account will be able to access the file. In the case multiple users access the Curator Tool from a single Windows account, the contents of the file are further encrypted using the Windows Data Protection API.

If at any time the credentials file is deleted or corrupted, the next time the user attempts to upload data, the configuration wizard will run automatically.

Building a Custom Dataview

By request, the Genesys Wizard has been designed to allow for additional pieces of information to be transferred to Genesys without having to modify the Wizard code. This is accomplished by modifying the dataview with the passport data.

Since dataviews are flat structures and the JSON structures sent to Genesys are hierarchical, a mapping approach was designed based on dataview column names and data types. The approach is as follows:

1. The following fields are required for any dataview and are not mapped directly to the Accession object:
 - a. *accession_id* – *Internal bookkeeping field*
 - b. *ACCENAME*, *OTHERNUMB* – *These fields are used for Accession Aliases which are separate API calls*
2. An underscore ('_') in a column name indicates that the data element is not at the root of the JSON hierarchy. For instance, an *accession* object has a *geo* child object with a *latitude* field. The name of the column would be *geo_latitude*. The approach supports hierarchy of any depth by including multiple underscores in the field name.

3. A dataview column can represent a delimited array of values. In order to signal the mapping engine that the column is an array, a '\$' character must be added to the end of the column name. For example, storage\$ is an array of storage ids representing the Type of Germplasm storage. The values within the dataview column must be delimited with a semicolon (;) character.
4. The data type returned from the dataview is important, especially for Boolean values. This signals the mapping engine how to serialize the value. For example, if a Boolean value is returned as the string '1', it will be sent to Genesys as a string '1'. Instead the dataview should include the statement *CAST(1 as bit)* in order to return a BIT field. The engine is able to recognize the data type and will serialize the value as *true*.