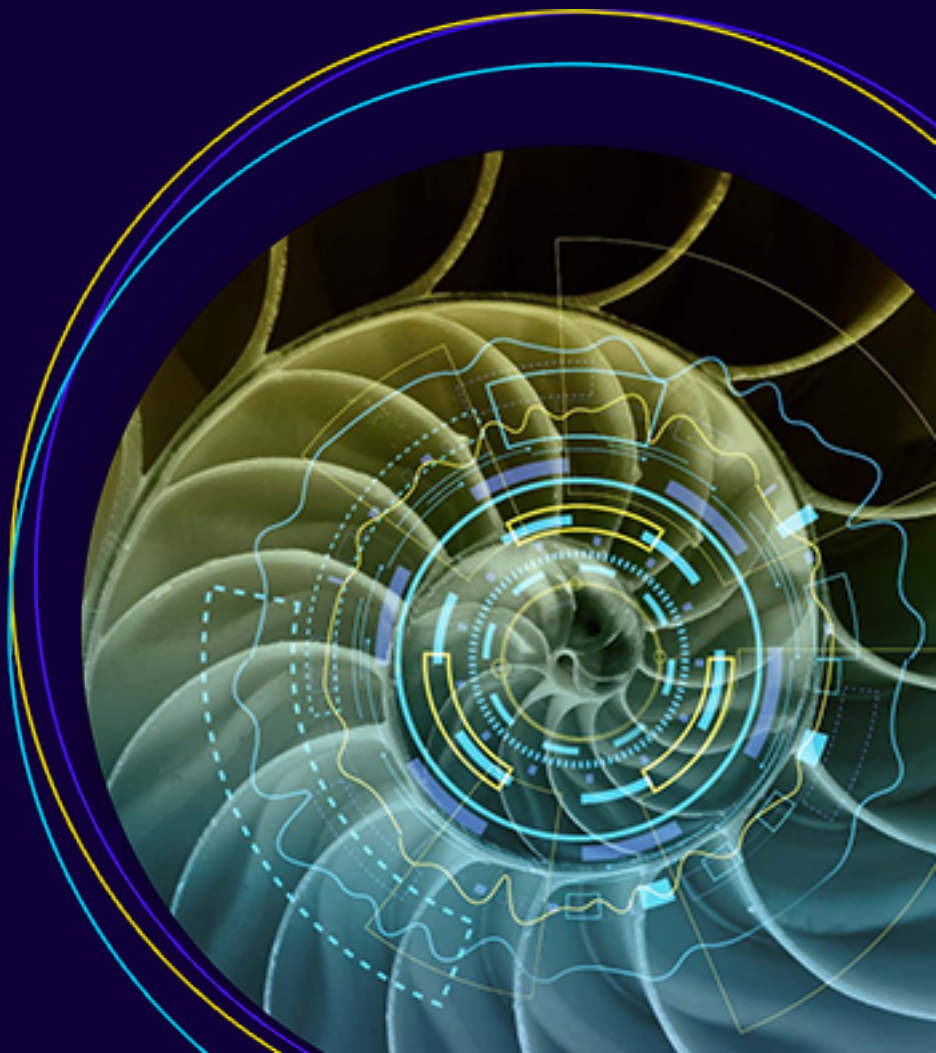
KRYON

# User Guide

## Kryon Web Service API v19.5.1





Document revision: 09-Feb-2020

# Contents

# CHAPTER 1: Introduction

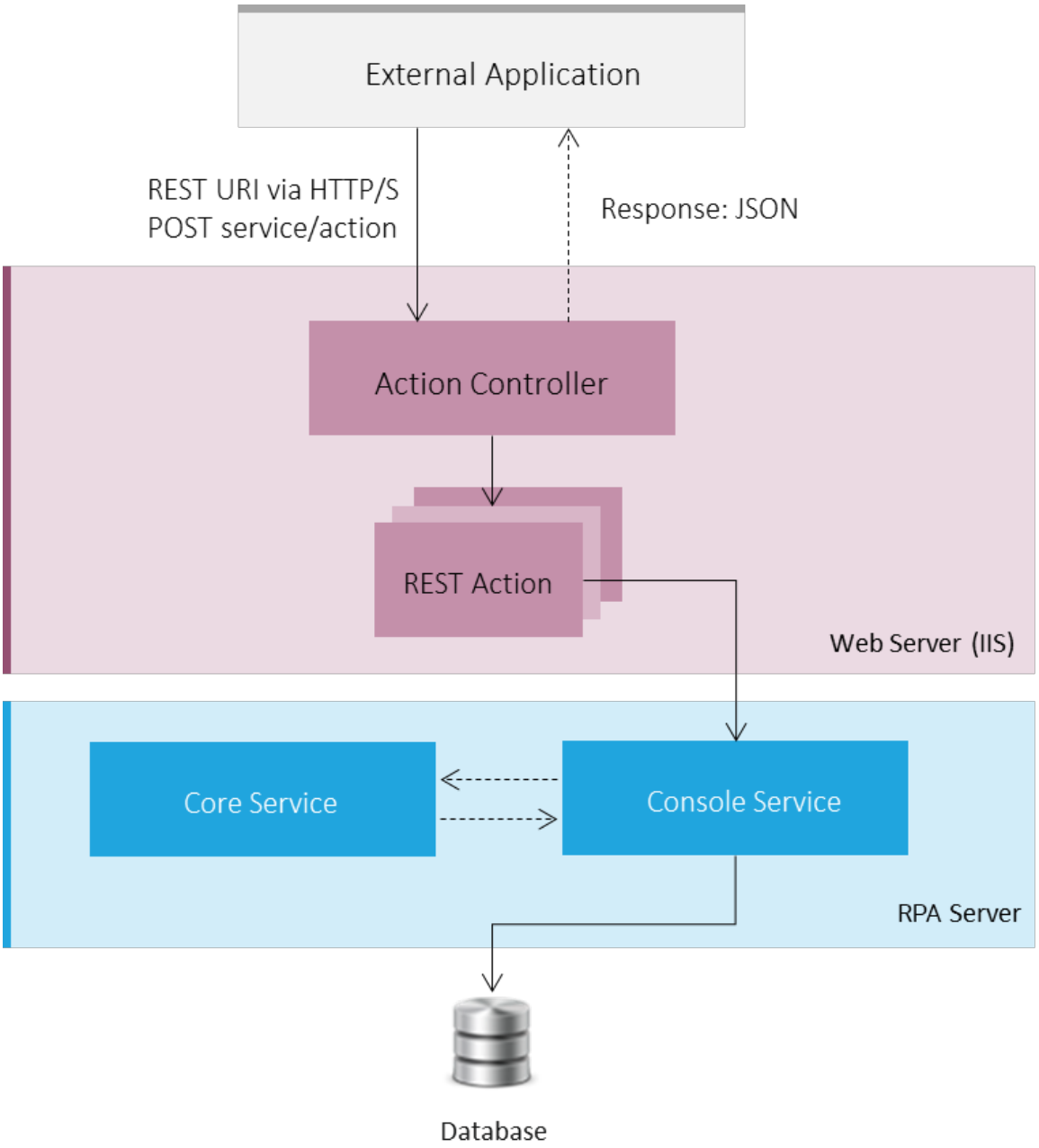
The Kryon Web Service API allows external applications to invoke/monitor the RPA tasks of the Kryon solution.

It utilizes the ASP.NET MVC (Web API) model with 2 methods:

- POST - creates a new task in the task queue
- GET - retrieves task status

All incoming requests are processed by the Kryon RPA Server and responses are sent back to the calling application. If a process fails, an error response is sent to the external application along with error details.

# CHAPTER 2: Architecture



# CHAPTER 3: Authentication

Each web request requires 2 custom entries in a header section:

- username - Kryon username
- pwd - Kryon password

Authentication is performed on the Kryon RPA Server for every API call, using the provided Kryon account credentials (username and password).

In case of authentication failure, the appropriate response, along with the error code is sent to the calling application.

| Code | Meaning |
|---|---|
| 0 | No Error |
| 1 | User Not Found |
| 2 | User Inactive |
| 3 | Server Error |
| 4 | License Error |
| 10 | Unknown |
| 11 | User Locked |
| 12 | Password Change Required |
| 13 | Password Expired |

# CHAPTER 4: API Reference

## Add Task

**POST** ConsoleServerAddress:API_port_number/*task/add*

- ConsoleServerAddress = FQDN of the Console Server as defined during Kryon RPA Server installation
- API_port_number = API port as defined during Kryon RPA Server installation
  - If required, you can check this number in IIS (in the Bindings for the WebAPI website)
  - Default value = 44445
- Example: MACHINENAME.COMPANYDOMAIN.COM:44445/task/add

| | |
|---|---|
| **API controller name** | Task |
| **Method name** | Add |
| **Input** | WebRequest (json) |
| **Output** | WebResponse (json) |

## Add Task: WebRequest Parameters

| | | |
|---|---|---|
| **WizardCustomId** | String | ID of the wizard to execute<br>• See Appendix A for information about viewing/creating wizard custom IDs |
| **NumberOfRuns** | Long | Number of wizard runs per task; default is 1 |
| **SingleRunEstimation** | Long | Time estimate for a single wizard run in minutes<br>*(This is used only for display in Kryon Console and does not affect the wizard run in any way.)* |
| **Variables** | Array (String, String) | List of initial variable names and values to be populated into the wizard |
| **QueuePriority** | Integer | Priority in task queue:<br>• 0 - normal (default)<br>• 1 - high |
| **GroupName** | String | Robot group to which to assign the task; default is empty (any available robot) |
| **MachineName** | String | Robot name to which to assign the task; default is empty (any available robot) |

## Add Task: WebResponse Parameters

```
{
TaskId: <long>,
Status: <int>,
Error: <int>,
OutputData: <string>
}
```

| | | |
|---|---|---|
| **TaskId** | Long | ID of the added task |
| **Status** | Integer | *(N/A – will be empty)* |
| **Error** | Integer | *(N/A – will be empty)* |
| **OutputData** | String | *(N/A – will be empty)* |

## Example

```
var request = {    WizardCustomid: 'A_123',
                   NumberOfRuns: 1,
                   SingleRunEstimation: 20,
                   Variables: [{Name: 'var1', Value: 'one'},
                           {Name: 'var2', Value: 'two'}]};
$.ajax({           url: '/task/add',
                   data: JSON.stringify(request),
                   type: 'post',
                   crossDomain: true,
                   contentType: 'application/json; charset=utf-
                   8',
                   beforeSend: function(request) {
                           request.setRequestHeader("username",
                           'username'),
                           request.setRequestHeader("pwd",
                           'password');
                   }
                   })
                   .done(function(resp) {
                           return resp;
                   })
                   .fail(function(error) {
                           throw new Error("Error getting the
                           data");
                   });
```

# Get Status

**GET** ConsoleServerAddress:API_port_number/task/status?tid=*taskid*

- ConsoleServerAddress = FQDN of the Console Server as defined during Kryon RPA Server installation
- API_port_number = API port as defined during Kryon RPA Server installation
  - If required, you can check this number in IIS (in the Bindings for the WebAPI website)
  - Default value = 44445
- Example: MACHINENAME.COMPANYDOMAIN.COM:44445/task/status?tid=taskid

| | |
|---|---|
| **API controller name** | Task |
| **Method name** | GetStatus |
| **Parameter** | TaskId (long) |
| **Output** | WebResponse (json) |

## Get Status: WebResponse Parameters

```
{
TaskId: <long>,
Status: <int>,
Error: <int>,
OutputData: <string>
}
```

| | | |
|---|---|---|
| **TaskId** | Long | ID of the queried task |
| **Status** | Integer | Task status *(see table below)* |
| **Error** | Integer | Error code *(see table below)* |
| **OutputData** | String | Output data reported by wizard<br>In order for this output parameter to be returned, the relevant wizard must utilize the Report Wizard Output advanced command.<br>• For additional details, see the document: ***Advanced Commands Reference Guide (Report Wizard Output)*** |

| Status Code | Meaning |
|---|---|
| 0 | Started |
| 1 | Stopped |
| 2 | Ended |
| 3 | Delayed |
| 4 | Inactive |
| 5 | Skipped |
| 6 | Queued |
| 7 | Faulty *(see table below)* |

| Error Code | Meaning |
|---|---|
| 0 | OK (no error) |
| 1 | General |
| 2 | Login |
| 3 | Expired |
| 4 | Failed To Create Task |
| 5 | Wizard Not Found |
| 6 | Failed To Get Status |
| 7 | Task Not Found |

## Example

```
$.ajax({     url: '/task/status?tid=1',
             type: 'get',
             crossDomain: true,
             beforeSend: function(request) {
                     request.setRequestHeader("username",
                     'username'),
                     request.setRequestHeader("pwd",
                     'password');}
             })
             .done(function(resp) {
                     return resp;
             })
             .fail(function(error) {
                     throw new Error("Error getting the data");
             });
```

# CHAPTER 5: Task Queue

The task creation process is managed by the Kryon Server according to robot availability.

If no robot is available to immediately execute a task, the task is created and placed in queue with the status **Queued**.

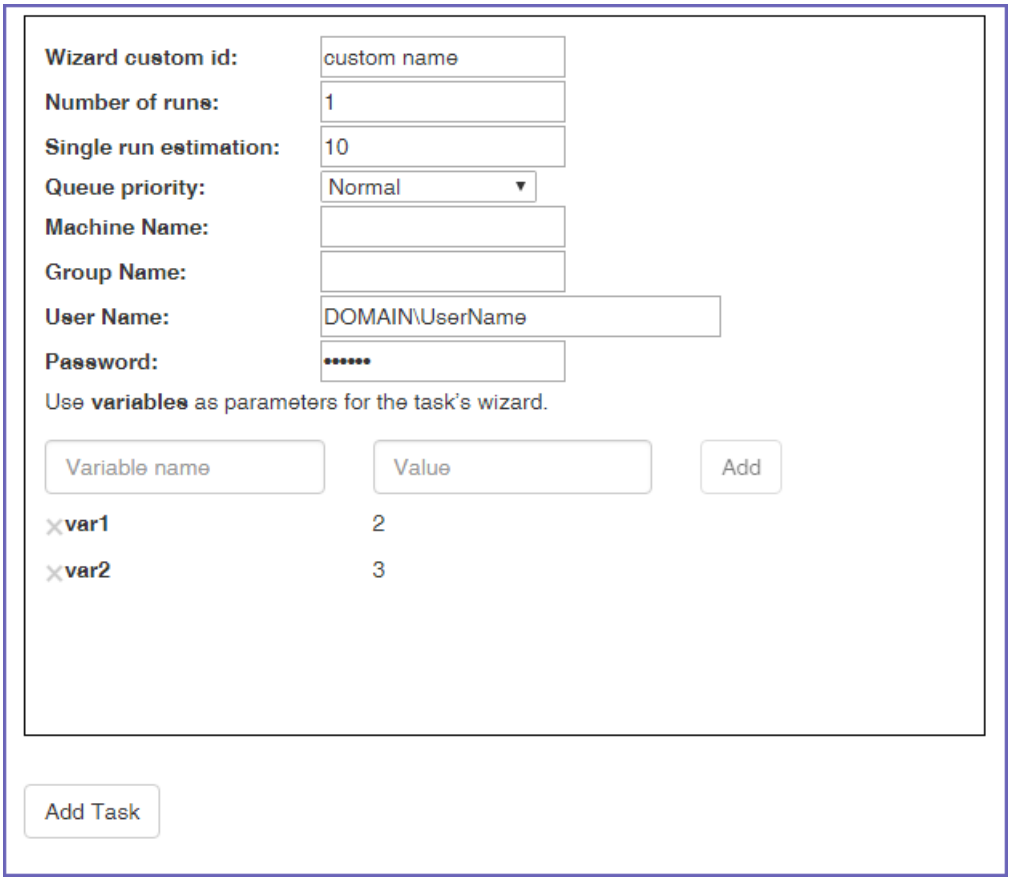
A queued task is saved for up to 24 hours. If no robot becomes available to perform the task during that time, the task status is changed to **Expired**, and the task is permanently removed from the queue.

# CHAPTER 6: Testing

The Web Service API can be tested using the provided **tester.html** file, which allows you to test the two API methods:

- The default location for this file is `{InstallFolder}\RPA\Kryon Web Server 64bit\WebAPI`

## Add Task

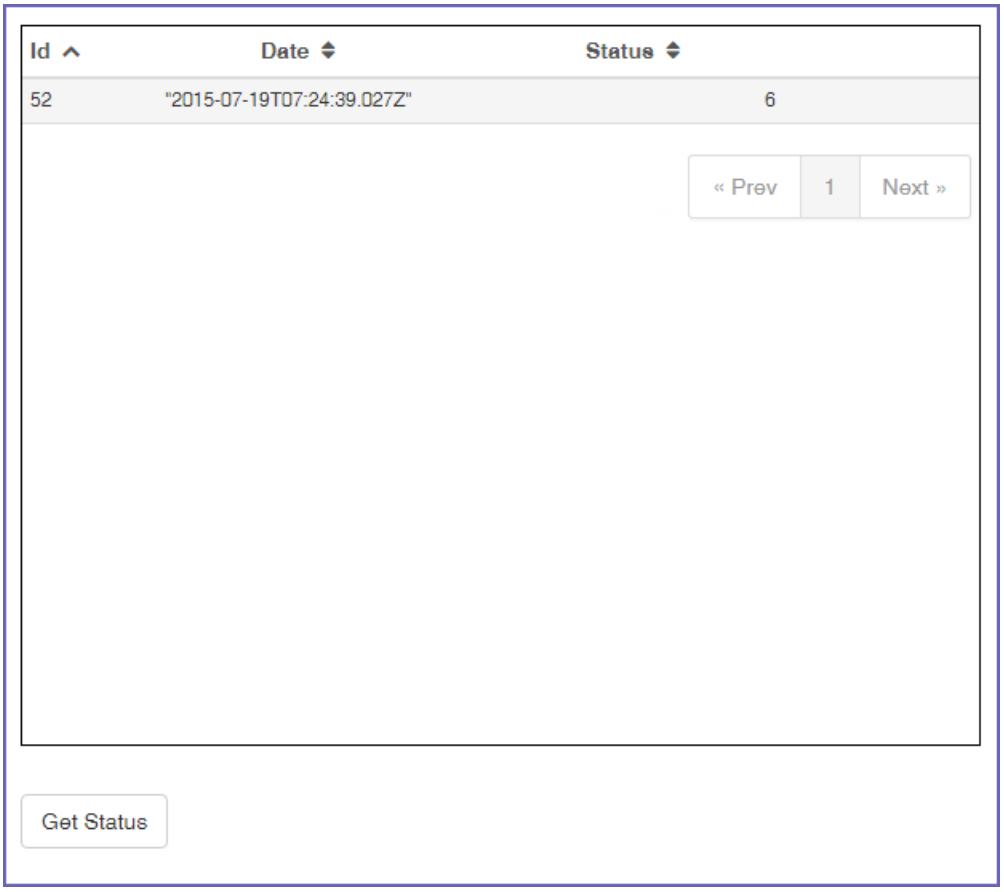



**NOTE**

Prior to testing, you should ensure that a wizard custom ID exists. See Appendix A for information about viewing/creating wizard custom IDs.

## Get Status

Use the tester to add task IDs for status monitoring:




**NOTE**

Upon manual page refresh (F5), task information is lost and can no longer be monitored from the tester (only from Kryon Console or via custom API implementation).

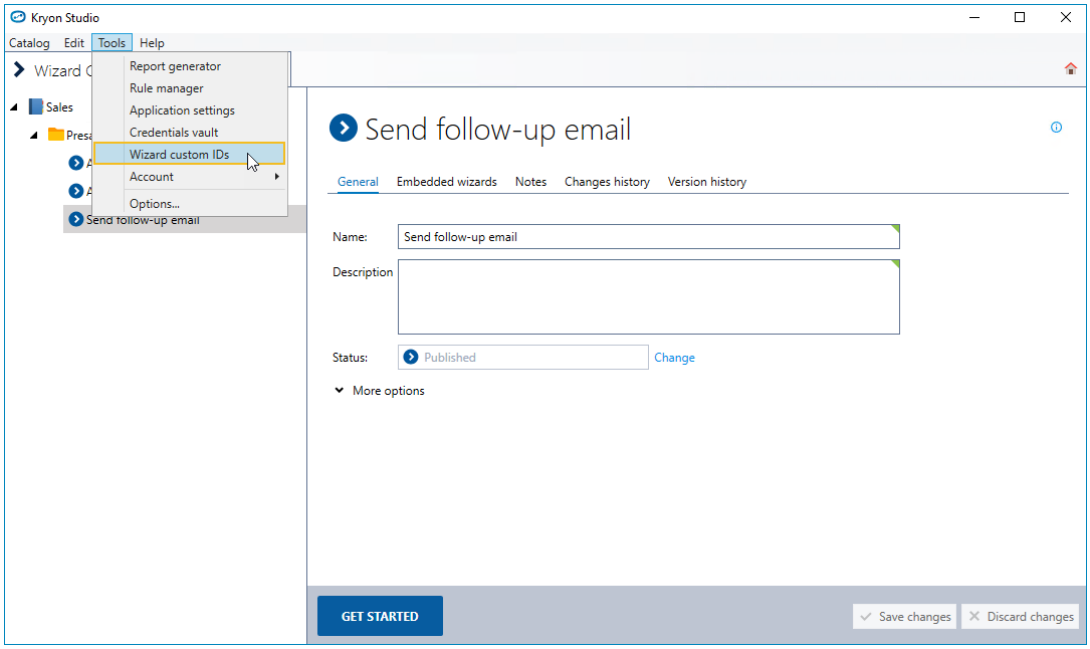
# APPENDIX A: Wizard Custom IDs

To add a task to the queue using the API, you must identify the wizard that the task will run. You do so by referencing a **wizard custom ID** assigned to the wizard in **Kryon Studio**.

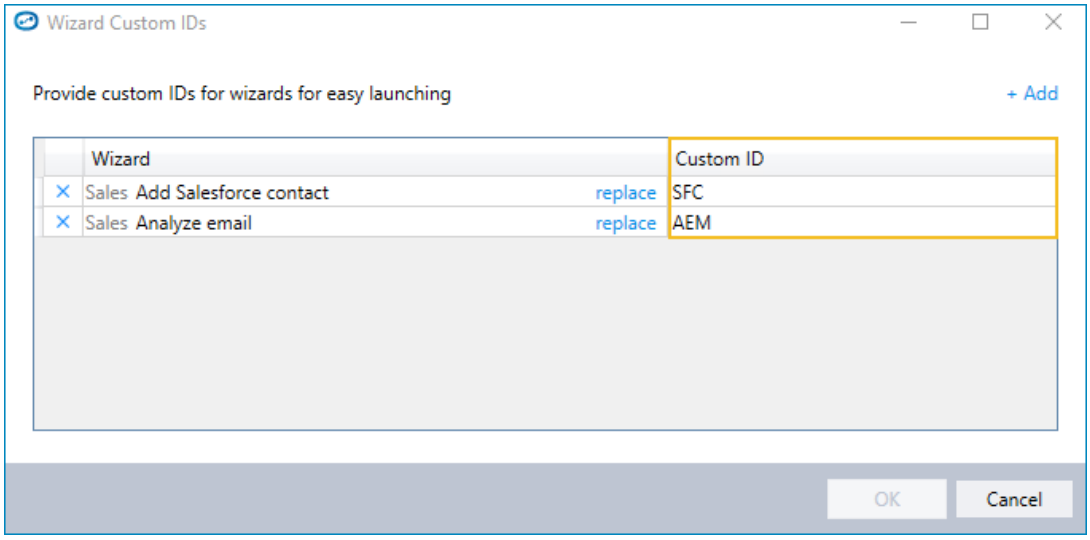
## Accessing wizard custom IDs

To view a list of wizard custom IDs:

1. From the menu in the main Kryon Studio window, click **Tools > Wizard custom IDs**




2. A list of all previously created wizard custom IDs will open:

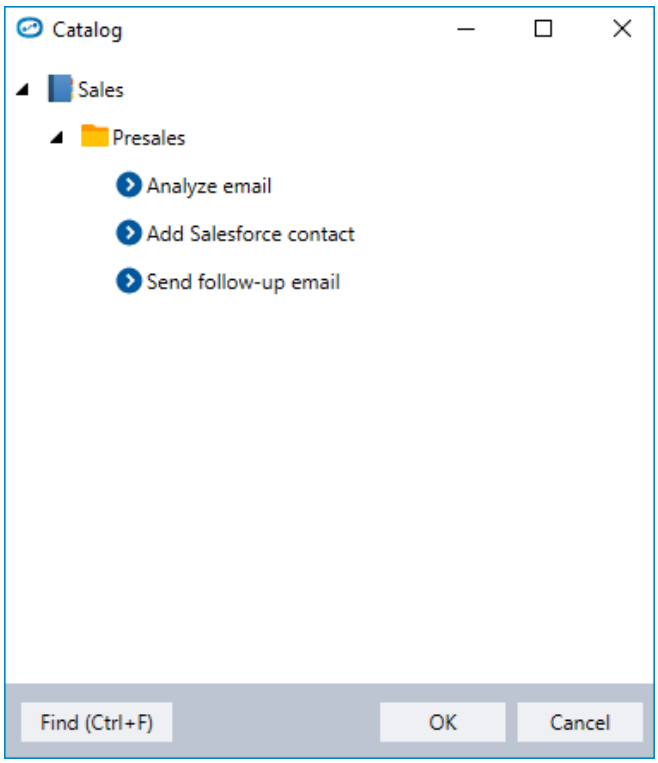


- If the wizard you need to reference appears in the list, you're in good shape. Simply make note of the value in the **Custom ID** column, and use it in the Add Task API call.
- If the wizard you are looking for doesn't appear in the list, you can create its wizard custom ID by following the steps below

## Creating a wizard custom ID

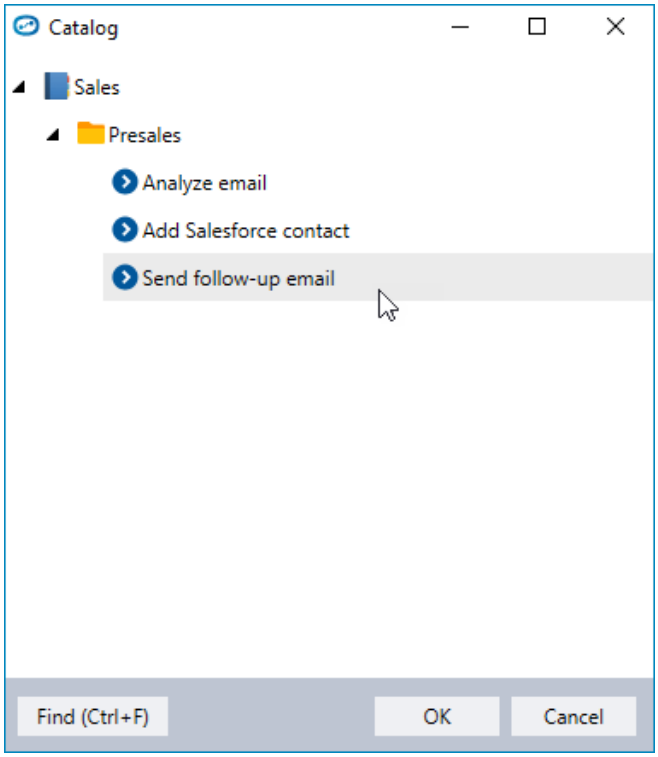
To create a new wizard custom ID:

1. Access the list of wizard custom IDs as described above
2. Click the + Add link in the upper right-hand corner of the window
3. A window displaying a hierarchy tree of all published wizards in the catalog will open

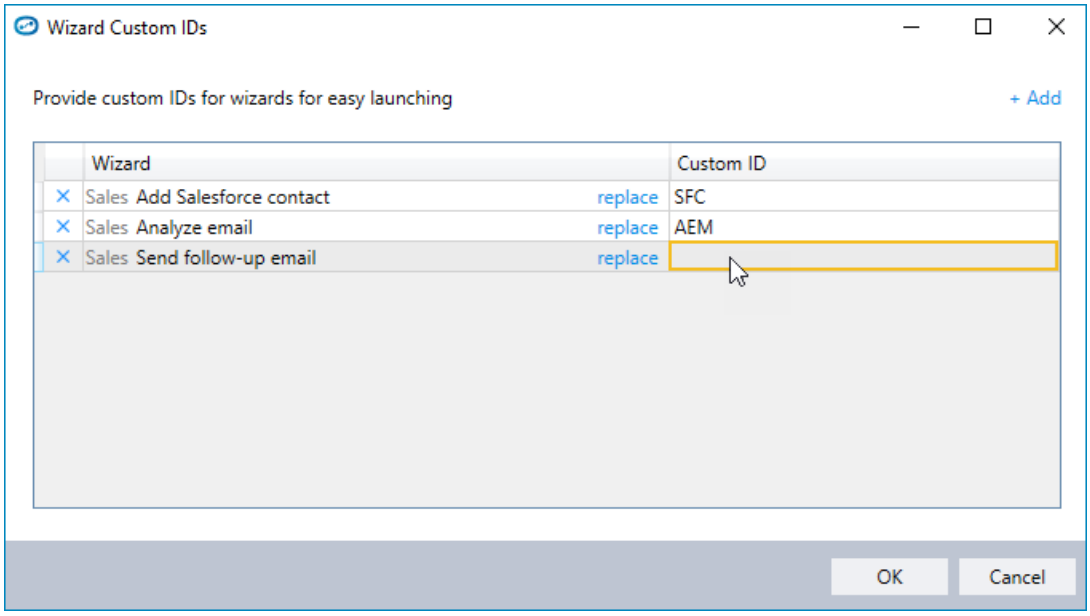



4. Locate the wizard you need either by browsing the tree or by clicking the Find (Ctrl+F) button to search for it

5. Select the wizard to which you want to assign a wizard custom ID, then click **OK**




6. The name of the wizard you selected will now appear in the list of wizard custom IDs:




7. Double-click in the empty **Custom ID** column for that wizard and type the desired ID
8. Click **OK** to save
9. Make note of the new wizard custom ID, and use it in the Add Task API call