GoPlant Data Integration for Historians, CMMS, LIMS and other external systems

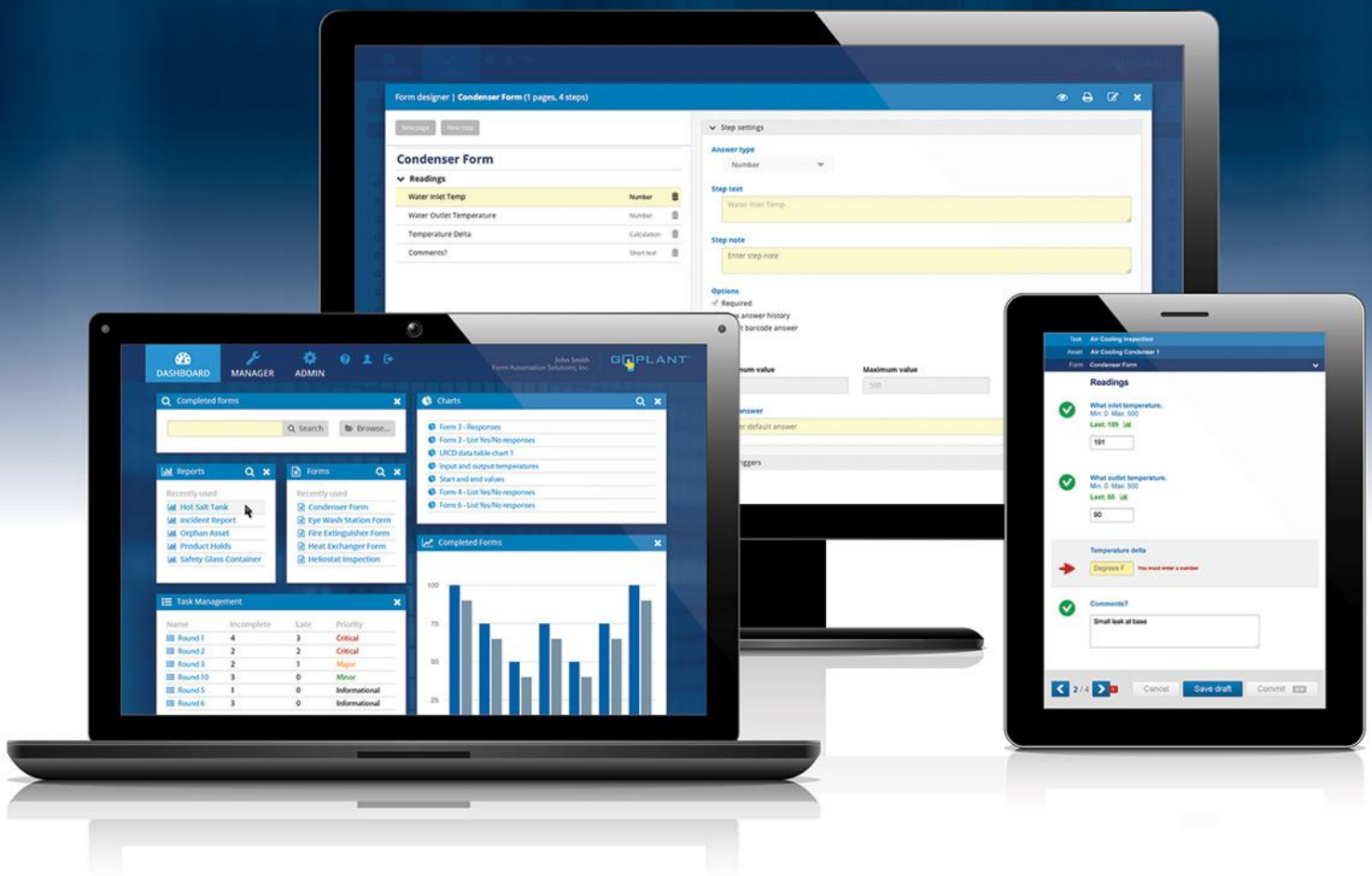

Table of Contents

Overview 3

Working with External Tags 3

External Tag Sources 4

Creating External data tags 4

GoPlant Reports 5

GoPlant Standard Reports 5

Custom Reports 7

GoPlant Application Programming Interface (API) 8

Overview 8

Examples 8

Database Reporting Table Queries 14

Overview

GoPlant enables data integration with external software systems through three primary methods:

1. **Scheduled File Output** – Goplant standard reports can be scheduled to run at regular intervals, delivering their output email or to a file folder location. Report output can be configured to return all responses, or only those which meet certain conditions. Those conditions include responses generating exceptions, responses associated with a particular asset, responses to tagged step questions, etc.
2. **Application Programming Interface (API)** - GoPlant provides a REST API to extract data. REST API endpoints provide data based on completed tasks, round, and forms. Additional endpoints provide all operator responses recorded in GoPlant as well as any exceptions generated by those responses. 3rd party software can use these endpoints to retrieve information from GoPlant.

   As with scheduled file output, API output can be limited to responses which generate exceptions, responses associated with a particular asset, responses to tagged step questions, etc.
3. **Database Reporting Table Queries (Premise Only) –** On premises installations of GoPlant may directly access the GoPlant database's reporting tables, enabling external systems to develop SQL queries using MS SQL reporting services or other database queries to extract any information required from the system. Any MS SQL-supported database connector(s) may be used to connect and query the GoPlant database. Output may contain data from individual tags or from entire storage groups within GoPlant.

Working with External Tags

External Tags are used to link GoPlant data points to input parameters in third party software. External Tags are created in GoPlant and then linked via a drag and drop interface to the appropriate Form step questions. This associates an Asset-Form answer with the appropriate label or “tag” in the other software system.

Typically, external tags are used to filter and map responses for consumption by Historian software such as OsiPi, Aspentech IP21, and Honeywell PHD. Tags are generally not required for mapping data to Enterprise Asset Management (EAM), Lab Information Management Systems (LIMS), and Water Information Management Systems (WIMS) software.

External Tag Sources

External Tags are grouped together in collections called External Tag Sources. There is no limit to the number of tags for each tag source. API calls may be filtered to only return data associated with a particular tag source. This filtering can be used to restrict the data exported to that required by the external software system.

Up to twenty (20) external tag sources may be defined. The Asset and Form pair create a unique link to the external tags or labels. This provides a robust means of linking your company's software systems with GoPlant information gathered in the field.

GoPlant comes preconfigured with Tag Sources for the following customer software systems:

- OSIsoft PI
- Honeywell Dynamo and PHD
- IBM Maximo
- JB Systems Mainsaver
- Aspen InfoPlus
- Hach WIMS.

A GoPlant instance can be configured to provide information for up to twenty different external software or database systems. The Tag Sources group together all the unique tags for a specific external software platform (such as OSI PI). This grouping allows the reports and REST API to retrieve all Tags associated with one external software system with one report or API call.

Creating External data tags

A form's step responses are associated with external software's data tags to corollate the GoPlant output to the external software's database structure. These relationships are created using GoPlant's external tag editor.

External tags are created within a GoPlant tag source (for example, OSIsoft PI system) to map these responses to an OSI PI tag. The external tags are the "label" or linkage that enables the software to query GoPlant for all information within a tag source and its multiple external tags.

External tags are also asset specific. Historian tags, Maintenance tags, Indicator tags, or anything required for external data systems may be added to a specific asset and form step.

Consider the following example. Three pumps of the same type are used in three different locations in a plant. Although all pumps use the same form in GoPlant, each pump can be associated with a unique external tag. This allows a single step question in the form to have a different external tag for each pump.

Once assigned to a step, external tags are used for CSV output, reports, or API calls to obtain information for other software systems.

GoPlant Reports

GoPlant Standard Reports

GoPlant has several built-in standard reports which are used to track work completed in the system. Of particular use for exporting data to 3rd party software is the standard report, Export Responses to CSV. This report generates CSV files that other software systems can consume. CSV output may also be used to export data for trending in Excel or other charting software.

The CSV file generated by this report includes the operator responses based on user-specified input controls. These input controls include:

- Storage Groups: Limits exported responses to those from forms in the specified storage group(s).
- Only Report Exceptions: If selected, only responses that triggered exceptions are exported.
- Filter by: Allows further filtering based on round, task, or tag source.
- Timeframe: Controls the interval for which responses are reported.

For each response that meets the specified criteria, the report generates a row of CSV output. There is an additional input control to specify which values are exported on each line.

For instance, consider a round, Example Round, in storage group, Plant One. To export the step text, response value, response date, and operator name for every response from this round over the past 72 hours, the input controls would be set as follows:

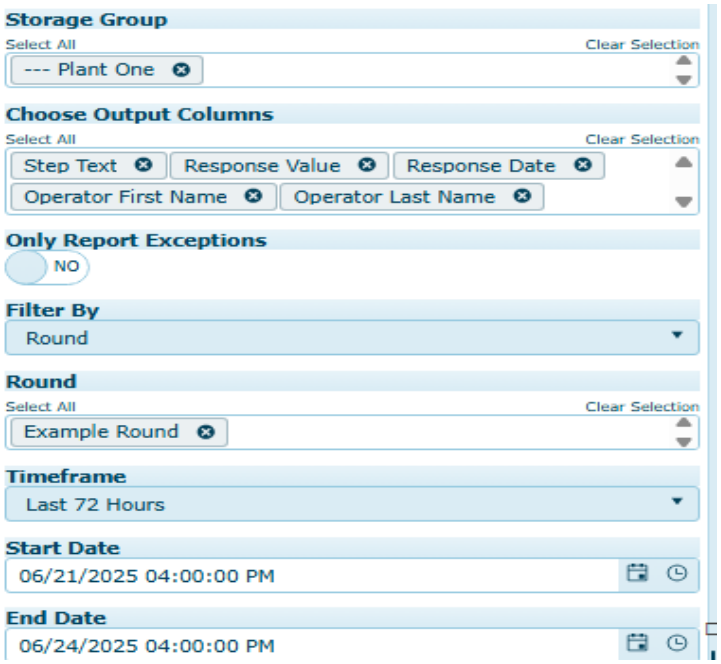

Figure 1: Export Filtered by Round

As another example, limit the responses to those that meet the following criteria:

- Tagged using tags from the tag source, OSIsoft PI system.
- Generated an exception
- Are from forms in storage groups Plant One and Plant Two
- Occurred over the last 24 hours

The columns to be generated for each response are:

- Storage group
- External tag
- Step text
- Response value
- Response date
- Form name

The input controls are set as follows:

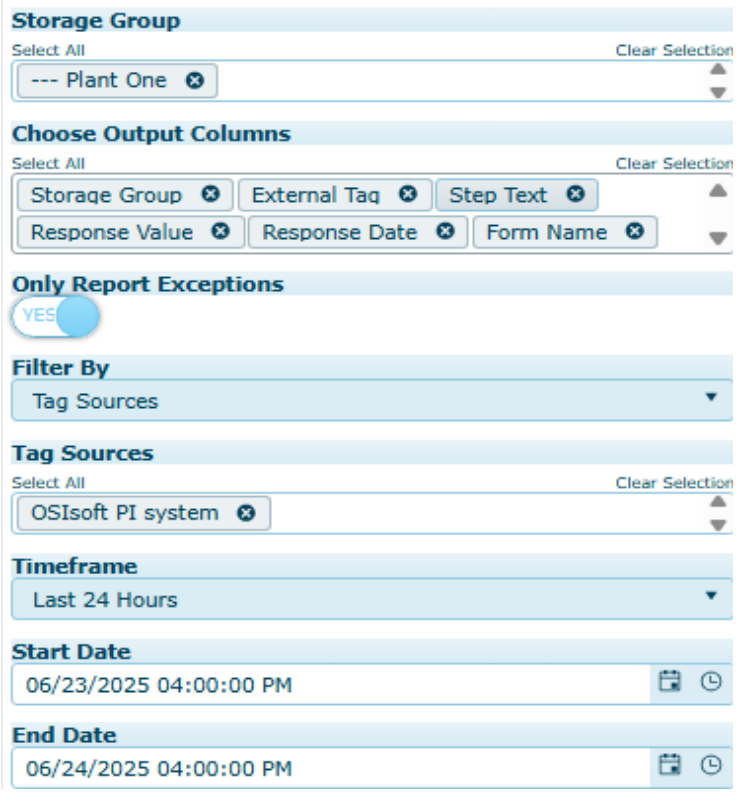

Figure 2: Export Tagged Exceptions

This report and all other GoPlant standard reports can be scheduled to run on a fixed schedule using GoPlant's report scheduling feature. Scheduled report output can be saved to a folder and periodically retrieved and processed by other software systems. Scheduled report output may also be sent via email to a list of users.

Custom Reports

GoPlant enables users to create custom Form Reports that include only the response data required. Any custom report created in GoPlant includes a CSV output option for both comma and tab delimited files. Custom reports allow end users to create their own files for use in Excel, MS Power BI, or other reporting tools.

Custom reports are not available for scheduling.

GoPlant Application Programming Interface (API)

Overview

GoPlant supports an extensive REST API which allows authorized users to track tasks, rounds, and forms as they are completed at your site. The API also allows users to view responses and exceptions. The rest API can be used to export data for use by third party software.

Comprehensive documentation of the new API is available online on your GoPlant instance. For instance, if your installation is https://mysite.goplant.mobi, your REST API documentation is available at https://mysite.goplant.mobi/api/docs

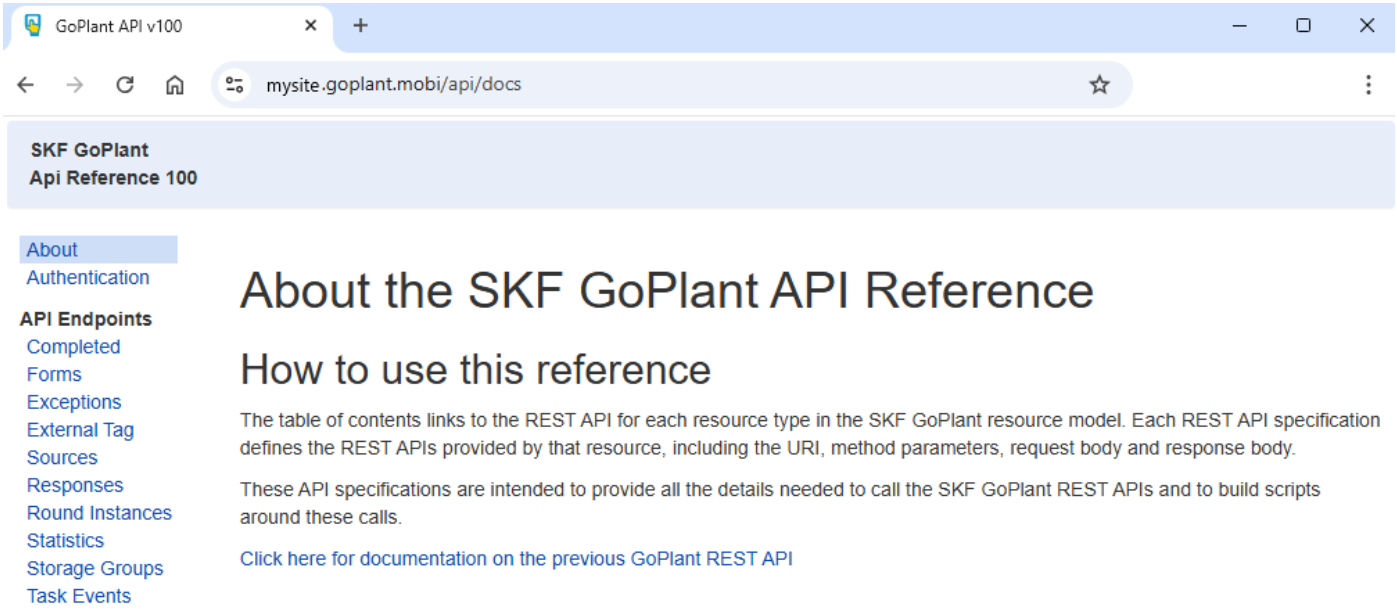

Figure 3: API Documentation Landing Page

Examples

The REST API can be used to collect information like that displayed in GoPlant's standard reports and widgets. In this example, we again want to see all responses to the a round, Example Round, in storage group, Plant One over the past 72 hours.

First, we place the following REST API call to retrieve all storage groups:

```
GET https://mysite.goplant.mobi/api/rest/storage-groups
```

The call returns a JSON record for each storage group on the system, including the following record for Storage Group Plant One:

```
{
      "uuid":  "5e22eb2b-16a7-47e9-a6ed-2f8bec6b8afd",
```

```
        "uri":  "/api/rest/storage-groups/5e22eb2b-16a7-47e9-a6ed-2f8bec6b8afd",
        "name":  "Plant One",
        "desc":  null,
        "storageGroupFullPath":  "Top Level / Plant One",
        "parentStorageGroupUuid":  "22944a8a-db66-445b-97c1-dc6131c52d4d",
        "parentStorageGroupUri":  "/api/rest/storage-groups/22944a8a-db66-445b-97c1-
dc6131c52d4d",
        "timeZone":  "Central Standard Time",
        "maxSessions":  25,
        "activeSessions":  0
}
```


We then query for all rounds instances in that storage group over the past 72 hours. In this example, the current date/time in UTC is 2025-11-06T17:00:00.

```
GET https://mysite.goplant.mobi/api/rest/round-instances? Start-Time=2025-11-
03T17:00:00&Storage-Group-UUID=5e22eb2b-16a7-47e9-a6ed-2f8bec6b8afd
```

Again, the call returns a JSON record for each round instance from that storage group completed over the past 24 hours. There is a single element associated with round Example Round:


```
{
          "uuid": "5a6a438e-120d-41de-abed-b6bc479c5364",
          "uri": "/api/rest/round-instances/5a6a438e-120d-41de-abed-b6bc479c5364",
          "roundName": "Example Round",
          "roundVariantName": "XXXX",
          "roundStorageGroupName": "Plant One",
          "roundStorageGroupUri": "/api/rest/storage-groups/5e22eb2b-16a7-47e9-a6ed-2f8bec6b8afd",
          "roundStartTimeUTC": "2026-01-06T20:33:28",
          "roundEndTimeUTC": "2026-01-06T20:33:37",
          "roundStatus": "Completed",
          "roundElementCount": 2,
          "completedElementCount": 2,
          "taskName": null,
          "taskEventUri": null,
          "roundReviewStatus": "No review
}
```


Place the following call to retrieve the round elements:

```
GET https://mysite.goplant.mobi/api/rest/round-instances/5a6a438e-120d-41de-abed-
b6bc479c5364/elements
```

The call returns both completed elements from the specified round instance:

```
{
    "count":  2,
    "roundElements":  [
         {
              "assetTag":  "Big Asset",
              "formName":  "All Exception Types",
              "status":  "Completed",
              "completedFormUri":  "/api/rest/completed-forms/a5148c00-c2c2-41cb-82eb-
7408be3b83c4",
              "startTimeUTC":  "2026-01-05T16:35:47",
              "endTimeUTC":  "2026-01-05T16:36:13"
          },
          {
               "assetTag":  "Little Asset",
               "formName":  "All Exception Types",
               "status":  "Completed",
               "completedFormUri":  "/api/rest/completed-forms/d5823701-2cb8-4f37-8823-
775d4a2a8738",
               "startTimeUTC":  "2026-01-05T16:36:16",
               "endTimeUTC":  "2026-01-05T16:36:32"
          }
     ]
}
```


Finally, extract the completedFormUUID from the completedFormUri fields and use those UUIDs to filter the responses to those two completed forms:

```
GET https://mysite.goplant.mobi/api/rest/responses?Completed-Form-UUID=a5148c00-c2c2-41cb-82eb-
7408be3b83c4,d5823701-2cb8-4f37-8823-775d4a2a8738
```

That call returns all eight responses to the two completed forms:


```
{
    "count":  8,
    "nextResponseUuid":  null,
    "responses":  [
        {
             "uuid":  "885c22f8-b183-4249-b225-f30cb58a0f3d",
             "uri":  "/api/rest/responses/885c22f8-b183-4249-b225-f30cb58a0f3d",
             "responseValue":  "I'm going on my own",
             "responseUnits":  null,
             "responseTimeUTC":  "2026-01-05T16:36:00",
             "responseModifiedTimeUTC":  null,
             "exceptionCount":  1,
             "stepType":  "Short text",
```

```
            "stepText":  "Tell me something",
            "userName":  "One Operator",
            "formName":  "All Exception Types",
            "formPageName":  "New Page",
            "completedFormUri":  "/api/rest/completed-forms/a5148c00-c2c2-41cb-82eb-
7408be3b83c4",
            "assetTag":  "Big Asset",
            "assetStatus":  "Disabled",
            "roundName":  "Example Round",
            "roundInstanceUri":  "/api/rest/round-instances/5a6a438e-120d-41de-abed-
b6bc479c5364",
            "taskName":  null,
            "taskEventUri":  null
        },
        {
             "uuid":  "9e994507-4dae-4b68-87e5-ebe87a296069",
             "uri":  "/api/rest/responses/9e994507-4dae-4b68-87e5-ebe87a296069",
             "responseValue":  "6",
             "responseUnits":  null,
             "responseTimeUTC":  "2026-01-05T16:36:01",
             "responseModifiedTimeUTC":  null,
             "exceptionCount":  2,
                   :
```


While this example may seem complicated, it is comprised of 4 API calls which can be scripted using PowerShell, Python, PowerBI etc.

Here is the same example written in PowerShell

```
$Website = "mysite.goplant.mobi"

# Prompt for username and password
[string]$userName = Read-Host -Prompt "Username"
[string]$userPassword = Read-Host -Prompt "Password"
$pair = "$($userName):$($userPassword)"
$encodedCreds = [System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes($pair))
$basicAuthValue = "Basic $encodedCreds"

$Headers = @{
   "X-Api-Version" = 100
   Authorization = $basicAuthValue
}

# Retrieve list of storage groups
$URL = "https://" + $Website + "/api/rest/Storage-Groups"
```

```
try {
    $response = Invoke-WebRequest -Method Get -Header $Headers -Uri $URL
} catch  {
    Write-Host "An unexpected error occurred: $($_.Exception.Message)"
}
# Find the uuid of storage group "Plant One"
$group_uuid = "-1"
$group_list = $response | ConvertFrom-Json
foreach ($group in $group_list.storageGroups) {
    if ($group.name -eq "Plant One") {
        $group_uuid = $group.uuid
        break
    }
}
if ($group_uuid -eq "-1") {
    Write-Host "Storage group not found"
    exit
}

# Query for all rounds in that storage group over the past 72 hours
$time_utc = [DateTime]::UtcNow.AddHours(-72)
$URL = "https://" + $Website + "/api/rest/round-instances?Start-Time=" + $time_utc + "&Storage-Group-UUID=" + $group_uuid
try {
    $response = Invoke-WebRequest -Method Get -Header $Headers -Uri $URL
} catch {
    # Handle any other unexpected errors
    Write-Host "An unexpected error occurred: $($_.Exception.Message)"
}
# Find the uuid of the first round instance named "Example Round"
$round_uuid = "-1"
$round_list = $response | ConvertFrom-Json
foreach ($round in $round_list.roundInstances) {
    if ($round.RoundName -eq "Example Round") {
        $round_uuid = $round.uuid
        break
    }
}
if ($round_uuid -eq "-1") {
    Write-Host "Round instance not found"
    exit
```

```
}

# Query for all the elements of that round instance
$URL = "https://" + $Website + "/api/rest/round-instances/" + $round_uuid +"/elements"
try {
   $response = Invoke-WebRequest -Method Get -Header $Headers -Uri $URL
} catch {
   # Handle any other unexpected errors
   Write-Host "An unexpected error occurred: $($_.Exception.Message)"
}

#Construct the list of Completed Form UUIDS to pass on the final call
$round_element_list = $response | ConvertFrom-Json
[bool]$first_element = $true
foreach ($round_element in $round_element_list.roundElements) {
   #Remove first 26 characters from URI to get UUID
   $element_uuid = $round_element.completedFormUri.Substring(26)
   if ($first_element -eq $true) {
      $Completed_Form_UUIDS = "?Completed-Form-UUID=" + $element_uuid
      $first_element = $false
   } else {
      $Completed_Form_UUIDS = $Completed_Form_UUIDS + "," + $element_uuid
   }
}
# Query for all of the responses associated with those completed forms
$URL = "https://" + $Website + "/api/rest/responses" + $Completed_Form_UUIDS
try {
   $response = Invoke-WebRequest -Method Get -Header $Headers -Uri $URL
   $response.Content | ConvertFrom-Json | ConvertTo-Json
} catch {
   # Handle any other unexpected errors
   Write-Host "An unexpected error occurred: $($_.Exception.Message)"
}
```

Database Reporting Table Queries

GoPlant contains “normalized” database tables that allow for easy and retrieval of all response data within GoPlant. The reporting tables are organized so that each row stores all answers to all Forms for all Rounds in GoPlant. Each step question answer is stored in a row within this table along with all associated Task, Round, Asset, User, and date time stamp information available.

Due to the normalization of the data each row contains repeated data for the Task, Round, Asset, and Form. This makes it simple to read anything required for that specific step question response.

All responses for a given Task, Round, or Form can be reported based upon any criteria available in the tables. Knowing the Form name, you can quickly pull all forms that were completed for a given date and time range.

Direct database query is only available for Premise customers due to security limitations. Database table detailed information along with data descriptions is available in our technical documentation.

Copyright Notice

Information contained in this document is proprietary to SKF USA and may be used or disclosed only with written permission from SKF USA. This document, or any part thereof, may not be reproduced without the prior written permission of SKF USA.

This document refers to numerous products by their trade names, in most, if not all, cases these designations are Trademarks or Registered Trademarks by their respective companies.

This document and the related software described in this manual are supplied under license or nondisclosure agreement and may be used or copied only in accordance with the terms of the agreement. The information in this document is subject to change without notice and does not represent a commitment on the part of SKF USA, Inc.

The names of companies and individuals used in examples in the manuals, and in any sample databases provided, are fictitious and are intended to illustrate the use of the software. Any resemblance to actual organizations or individuals, whether past or present, is purely coincidental.