PRUVAN

# Integrating with the Pruvan API (version 2)

# Integrating with the Pruvan API (version 2)

## Table of Contents

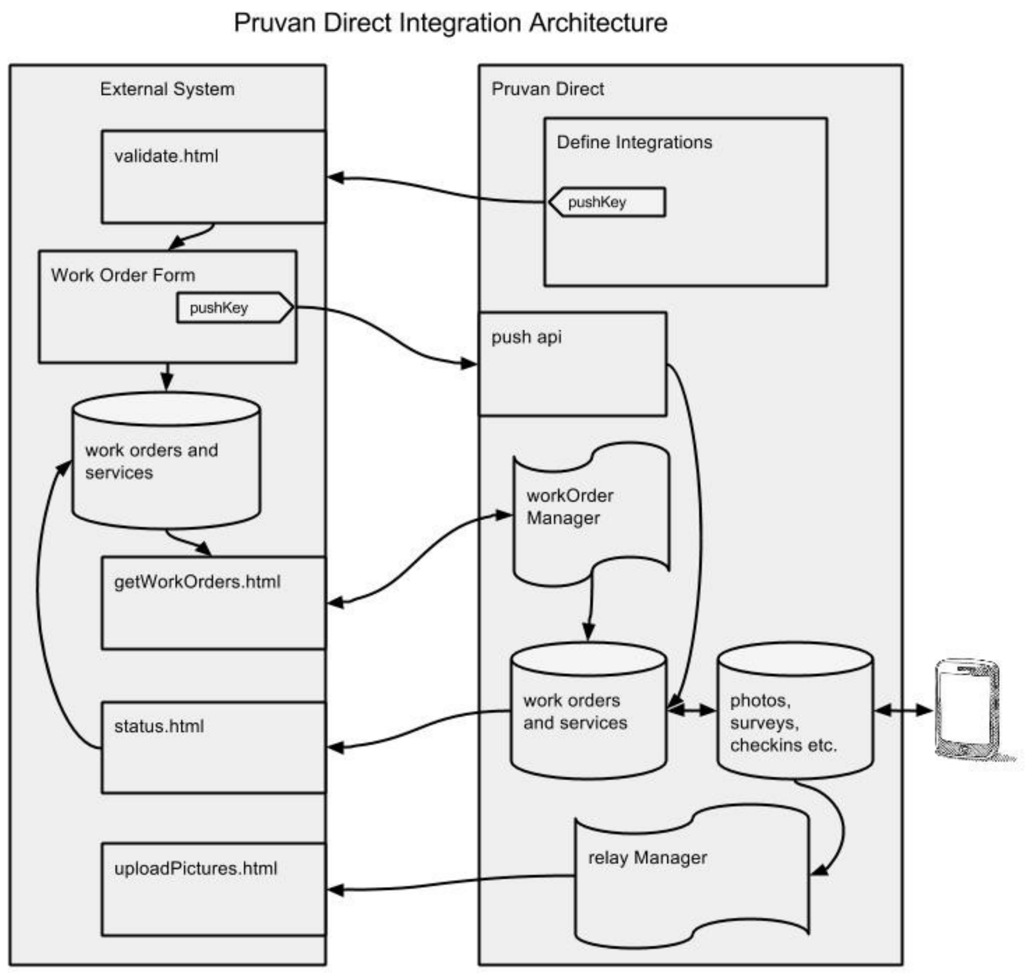
# Integrating with the Pruvan API (version 2)

## Overview

With the Pruvan API it's easy to get your work order management system integrated with Pruvan Direct.

This document covers the Pruvan API version 2. Version 2 of the Pruvan API makes it easier than ever to create a tight integration with Pruvan Direct and your system.

## Architecture



# How it works

Integration with Pruvan Direct through the Pruvan API requires you to implement the following 4 web service endpoints:

| Web Service Endpoint | Purpose |
|---|---|
| **validate** | Interface to authenticate and exchange user credentials |
| **getWorkOrders** | Interface to pull work orders for authenticated users into Pruvan |
| **uploadPictures** | Interface to push pictures and survey/mobile form data to you |
| **status** | Interface to push work order status signals from Pruvan to you |

## Flow

An external system implements the 4 required Pruvan direct web services (validate, getWorkOrders, status and uploadPictures). The integration is setup via the Define Integrations form in the Pruvan webUI. The validate web service is used to validate the user's access credentials on the External System and, upon success, transfers the Pruvan puskKey to the External System.

Once validated the Pruvan Direct user can request a work order import from the External System to Pruvan via the Integrations Form. The Pruvan WorkOrder Manager handles this asynchronous request and logs progress for the user. Upon successful completion, the user's work orders are available in the Pruvan webUI's Projects form.

The Pruvan user can assign the work orders to his subusers. Work orders can then be completed in the field and photos, surveys and checkins (work items) can be uploaded from the mobile device for QA in Pruvan.

Once the work items are QA approved in Pruvan the work items are uploaded to the External System via the Pruvan Relay Manager. The Relay Manager sends the work items to the External System's uploadPictures web service.

Users can refresh work order information from the External System via the Pruvan webUI manually. Users can also receive real time updates of work order changes from the External System via the Pruvan Push API.

The Pruvan PushAPI works by providing the external system a secure and unique url that can be used to send work order updates to Pruvan in real time (see the Pruvan Push API section later in this document for more details).

### A note about naming

The Pruvan API uses the nomenclature of work orders and services for Pruvan Direct projects and tasks. A Pruvan API work order is exactly the same as a Pruvan Direct project and a Pruvan API service is exactly the same as a Pruvan Direct task. The nomenclature difference is due to Pruvan Direct's relationship to the Pruvan Enterprise product where work orders and services names are configurable. The above terms are used interchangeably.

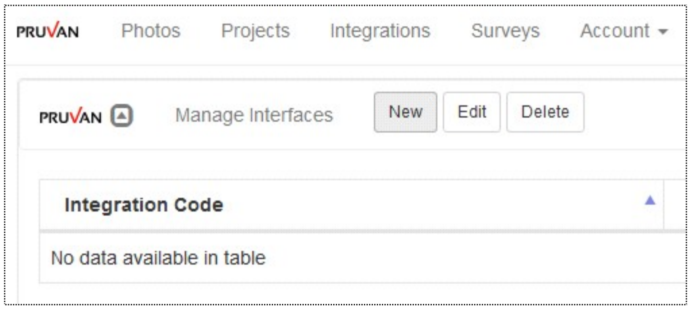
## Audience

The audience for this document is a systems architect or a developer that is tasked with integrating their system with Pruvan.  You should have a good understanding of web based integration technologies and data formats.

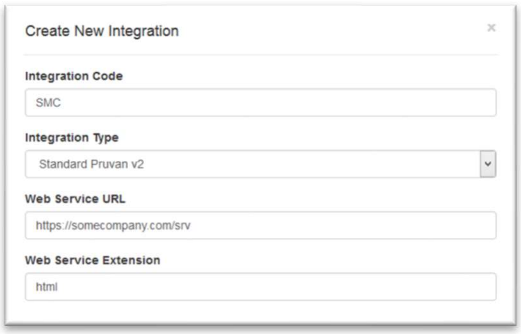
## Getting Started

To get started you should create a Pruvan Standard v2 Integration in your Pruvan Direct Account.

1. Login to you Pruvan account at www.direct.pruvan.com and select Integrations from the main menu.



2. Click "New" to create a new Integration
3. Enter an Integration Code and select "Pruvan APIv2" as the type
   - Internal or External depending on your needs
4. Enter the url for your implementation of the Pruvan API.
5. Enter your web service extension. The default is html but you can change this to whatever is required for you web service technology (.php, .svc or nothing).
6. The web service calls will be of the form: {web service url}/{web service name}.{web service extension).



Create New Integration

Integration Code: SMC

Integration Type: Standard Pruvan v2

Web Service URL: https://somecompany.com/srv

Web Service Extension: html

7. Enter a username password.
   Click on "Test" to send the validate JSON to the validate web service. This would be **https://somecompany.com/srv/validate.html** in the above example. You will need to have your validate web service up and running for this to work.

Getting the integration set up will allow you to test and fully develop your Pruvan integration.

# Developing your Pruvan API web services

## Pruvan API Endpoints

The Pruvan API is JSON based. Each Pruvan API web service endpoint receives and returns a specific JSON payload.

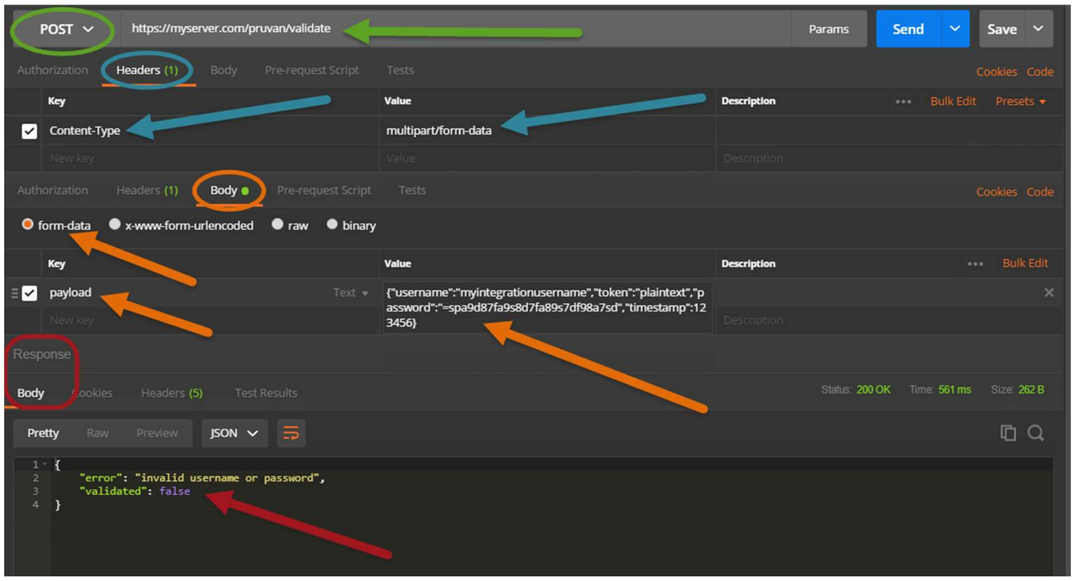
| **Web Service Endpoint** | **Purpose** |
|---|---|
| validate | Interface to authenticate and exchange user credentials |
| getWorkOrders | Interface to pull work orders for authenticated users into Pruvan |
| uploadPictures | Interface to push pictures and survey/mobile form data to you |
| status | Interface to push work order status signals from Pruvan to you |

## URL format

The web service calls are of the form {web service url}/{web service name}.{web service extension} as defined in the Create/Edit Integrations form. For example **https://somecompany.com/validate.html** would be called for validate with **https://somecompany.com** defined as the web service url and html was the web service extension. Leaving the extension blank would call **https://somecompany.com/validate**. All POSTs from Pruvan will have the Content-Type header: Mulipart/Form-Data. HTTPS is required, HTTP requests are not supported.

An example test to your /validate endpoint using Postman would look like the following:

- A POST to ***your*** endpoint – Green
- A Content-Type: multipart/form-data Header – Blue
- A Form-Data style body with the Key (field name) *payload* and the value containing a JSON object – Orange
- A Response, from ***your system***, containing a JSON object with the Header Content-Type: text/plain or application/json - Red



https://myserver.com/pruvan/validate

| Key | Value | Description |
|---|---|---|
| Content-Type | multipart/form-data | |
| New key | Value | Description |

| Key | | Value | Description |
|---|---|---|---|
| payload | Text | {"username":"myintegrationusername","token":"plaintext","password":"=spa9d87fa9s8d7fa89s7df98a7sd","timestamp":123456} | |
| New key | | | Description |

Body

Status: 200 OK Time: 561 ms Size: 262 B

```
{
    "error": "invalid username or password",
    "validated": false
}
```

# The validate Web Service

The validate web service will receive the JSON payload below when the "Validate" button is clicked on the Create/Edit Integrations form. It will be sent to {web service url}/validate.{web service extension} in the POST variable "payload":

```
{
 "username": "someuser",
 "password": "d28069c84398483874d2cba15f10ffdd996ec6f5",
 "token": "dda25c60-4fce-11e4-916c-0800200c9a66",
 "pushkey": "https://direct.pruvan.com/api/?apiKey=Qc71... "
}
```

## Authentication

Passwords are sent in two ways. The **password** field contains a sha1 hash of the password that the user entered in the integration setup. When using an https url (required) the **token** field contains the plaintext version of the password that the user entered. *We suggest that you provide your users with an API key for the integration password* and validate this API key using the **token** field. The API key (often a UUID) should be a password that works with your APIs but is not valid for use as an interactive login.

Note that the Pruvan pushkey is sent with the payload of validate. This allows you to automatically retrieve and store the Pruvan Push Key for your users. If you wish to support Pruvan Push you should store the push key in your user record and use it to send push updates for that user's orders. See the Pruvan Push API section later in this document.

If the username and password are valid, you should return the following as a JSON string:

```
{
 "error":"",
 "validated":true
}
```

If the username and password are invalid, you should return the following as a JSON string:

```
{
 "error": "invalid username or password",
 "validated":false
}
```

# The getWorkOrders Web Service

The getWorkOrders web service call is initiated from Pruvan. You respond to this call with a JSON object that contains the work orders you wish to load into Pruvan for a specified user. The getWorkOrders web service is a "pull" request from Pruvan to your system. Pruvan also supports "push" requests from your system through the Pruvan Push API. See the Pruvan Push API section later in this document for more details.

## getWorkOrders web service call

The getWorkOrders web service will receive the JSON payload below when the "Import Orders" button is clicked on the Integrations form. It will be sent to {web service url}/getWorkOrders.{web service extension} in the POST variable "payload":

```
{
 "username": "someuser",
 "password": "d28069c84398483874d2cba15f10ffdd996ec6f5",
 "token": "dda25c60-4fce-11e4-916c-0800200c9a66",
 "timestamp": 1393278236
}
```

## getWorkOrders Modes

The getWorkOrders web service supports **incremental** and **full** modes for synchronizing/refreshing work orders. To initiate a full refresh click "Full Refresh" on the Create/Edit Integrations form. To initiate an incremental refresh click "Import Orders" on the Integrations form.

In incremental mode the **timestamp** field will be set to a valid timestamp and you should only return orders that have been changed or created since that time for the specified username. In full refresh mode the **timestamp** will be set to null and you should return all current work orders for specified username.

## getWorkOrders JSON object

If the user is valid you should respond to getWorkOrders with an array of Pruvan workOrder JSON objects. The simplest form of a Pruvan workOrder creation is this:

```
{
   "workOrders": [
   {
      "workOrderNumber": "Simple",
      "address1": "110 East Main Street",
      "city": "Round Rock",
      "state": "TX",
      "zip": "78664",
      "services": [{"serviceName": "Task"}]
   }
  ],
 "error": ""
}
```

Note that the workOrder objects are returned in a workOrders array. The simple example above represents the minimum required set of fields for a valid Pruvan workOrder object.

## API details

If a getWorkOrders request is made and there are no updates to send simply return an empty workOrders array:

```
{"workOrders": [],"error": ""}
```

If the username and password are invalid or there is an exception, you should return the following as a JSON string with an appropriate error message:

```
{"error":"error message"}
```

## Available workOrder Object Fields

The workOrder object allows you to specify most work order/project properties in Pruvan. Once a work order is created only updatable fields can be updated.

| Data Element | Required | Updateable | Default | Description |
|---|---|---|---|---|
| **workOrderNumber** | Y | N | | Work Order/Project Number; this will be displayed in Pruvan as "{client code}::workOrderNumber". |
| **workOrderInfo** | N | Y | | Work Order information that is displayed on the work order details screen on Pruvan Mobile. Useful for lock box codes etc. |
| **address1** | Y | N | | Address line 1 |
| **address2** | N | N | | Address line 2 |
| **city** | Y | N | | City |
| **state** | Y | N | | State |
| **zip** | Y | N | | Zip |
| **country** | N | N | | Country |
| **assignedTo** | N | Y* | | Pruvan subuser to assign to. To assign to all subusers on an account set assignedTo: "~all~". *Internal integrations can directly update, external integrations can indirectly update - see Pruvan Integration Types below. |
| **status** | N | Y* | Un-assigned | Pruvan status. *Internal integrations can directly update, external integrations can indirectly update - see Pruvan Integration Types below. |
| **dueDate** | N | Y* | Today | Field due date. *Internal integrations can directly update, external integrations can indirectly update - see Pruvan Integration Types below. See more info in the Date Fields section. |
| **instructions** | N | Y* | | Instructions for the field - viewable on device. These are displayed as 'webviews'. This field can contain light HTML formatting to include: headers, text styling (bold, etc), line breaks, href links, etc. An example of what won't work is embedded video. *Internal integrations can directly update, external integrations can indirectly update - see Pruvan Integration Types below. |
| **clientStatus** | N | Y | | Client order status – see status mapping. |
| **clientDueDate** | N | Y | | Client due date. See more info in the Date Fields section. |
| **clientInstructions** | N | Y | | Client instructions. |
| **description** | N | N | | Description; only viewable in Pruvan Project Management tab – cannot be updated by the integrations. |
| **reference** | N | Y* | | Reference; only viewable in Pruvan Project Management tab. *Internal integrations can directly update, external integrations can indirectly update - see Pruvan Integration Types below. |
| **attribute7-15** | N | N | | Optional attributes |
| **gpsLatitude / gpsLongitude** | N | N | | GPS Lat/Long are used to specify the property address. If not provided Pruvan will geo-locate the address. |
| **options** | N | N | | A JSON string for Mobile Configuration Options set at the Work Order level. See options section for details. |
| **startDate** | N | Y | | Setting a startDate will prevent the project from opening on Pruvan Mobile before the startDate. |
| **source_wo_id** | N | N | | The work order ID as determined by the source work order |

| | | | | |
|---|---|---|---|---|
| | | | | provider |
| **source_wo_number** | N | N | | The work order number as determined by the source work order provider |
| **source_wo_provider** | N | N | | The (ultimate) source work order provider (the Tier one provider) |
| **services** | Y | Y | | An object containing the work order's services ; at least one service/task is required on create. Services are not required for updates. If services are included on an update ALL the services must be included. Missing services will be deleted. |

## Available workOrder service Object Fields

The workOrder object's service element allows you to specify service/task properties in Pruvan. Once a service is created it cannot be updated. Services can be deleted/added to an existing work order by sending a new complete services payload (see the example below). You cannot currently update a survey template ID by updating the services element. If the serviceName is the same as the existing one then the template ID will not update. You must either make 2 calls; one to delete the survey service and one to create it again with the new survey template ID, or change the serviceName of the service with the new template ID.

| Data Element | Required | Update | Description |
|---|---|---|---|
| **serviceName** | Y | N | Service/task name |
| **survey** | N | N | Survey template ID, if just the templateID is given it will pull the latest published template from the local Pruvan account. If a version number is passed it will load that specific version of the template (must be published). To load a survey from a separate account use this format: "{master_username}::templateID" |
| **instructions** | N | Y | Task Instructions; visible on device. These are displayed as 'webviews', see instructions for the work order level. |
| **options** | N | Y | A JSON string for Mobile Configuration Options set at the Service level. See options section for details. |
| **source_service** | N | N | The service as determined by the source work order provider |
| **source_service_id** | N | N | The service ID as determined by the source work order provider |
| **attribute1-15** | N | N | Optional attributes |

## Numbered Attributes

Numbered attributes (ie attribute7) will follow a Work Order, or Service, and attach themselves to any resultant objects; such as pictures, CSRs, and Surveys. Attributes 1-6 are reserved for the Work Order address fields. Work Order attributes 7-15, and Service attributes 1-15, are available for internal reference data. If you assign attributes 1-6 at the Work Order level they will be over-written.

## Updates

Fields identified as updatable in workOrder object reference above can be updated. The minimal update is the workOrderNumber and a single field.

The services element is not required in an update. If the services element is included in an update it is considered a full replacement. This means that if you have service1 and service2 on a workOrder and you send an update with only service1 – service2 will be deleted. No fields on a service are updatable.

### Requirements

The workOrderNumber must be unique. Duplicate workOrderNumbers will be rejected.

### Limits

A maximum of 5000 work orders can be retrieved in a single call. If you have the need to load more than this limit the Push API can be used to synchronously load orders, n orders at a time.

## The uploadPictures Web Service

The uploadPictures web service call is initiated from Pruvan when photos, surveys or checkins are published. The uploadPictures web service is a "push" request from Pruvan to your system.

This request will contain metadata for photos, surveys and checkins. The uploadPictures JSON object will be the same basic structure for all file types. Surveys and checkins have additional elements for survey and checkin details respectively.

Videos will be sent as pictures and contain a jpg of a still taken from the video. Videos will be identifiable by a link in the "videoUri" key. For all other types, this key will be null. To retrieve the video file use GET on the uri in the videoUri field. The link will expire 24hrs after it was sent. To get a new link you must re-publish the video to your integration.

Pruvan will send the picture, survey, file, or checkin file as multipart/form-data in the **file** field.

### uploadPictures web service call

The uploadPictures web service will send the JSON payload below when the items are published. It will be sent to {web service url}/uploadPictures.{web service extension} in the POST variable "payload":

```
{
  "username": "someuser",
  "password": "d28069c84398483874d2cba15f10ffdd996ec6f5",
  "token": "dda25c60-4fce-11e4-916c-0800200c9a66",
  "key2": "50010100",
  "key4": "Field QC Inspection",
  "attribute1": "110 East Main Street",
  "attribute2": "110 East Main Street",
  "attribute3": "Round Rock",
  "attribute4": "TX",
  "attribute5": "78664",
  "attribute6": "USA",
  "attribute{7-30}": "",
  "notes": null,
  "authenticated": "1",
  "csrCertifiedLocation": "1",
  "csrCertifiedTime": "1",
  "csrLocationSource": "mobile",
  "csrPictureCount": null,
  "csrTimeStampSource": "mobile",
  "locationDifference": "24",
  "gpsAccuracy": "10.0",
  "gpsLatitude": "30.50881246",
  "gpsLongitude": "-97.67852886",
  "gpsTimestamp": "1414434574",
  "creationDate": "2014-10-27 13:29:35",
  "evidenceType": "Survey",
  "fileType": "picture",
  "template": "vendor1::PRVQCINSP-v1",
  "uploaderVersion": "mobile 3.12.4",
  "uuId": "fc22d86b-b200-4375-8689-f582ae9bc0d9",
  "workDay": "2014-10-27",
  "timestamp": 1414434590
  …. (additional fields passed – please see uploadPcitures object reference)
}
```

If the username and password are valid, you should return the following as a JSON string:

```
{"status":true,"error":null}
```

If there's an exception, you should return the following as a JSON string with an appropriate error message:

```
{"status":false,"error":"error msg"}
```

## Available uploadPictures Object Fields

The uploadPictures object provides you with all available data for pictures, surveys, files, and checkins in Pruvan.

| Data Element | Description |
|---|---|
| **username** | The username provided by the user for authentication. |
| **password** | The sha1 hashed password provided by the user for authentication. |
| **token** | The plain-text password provided by the user for authentication (https only). |
| **pictureId** | The picture ID as it is stored in the Pruvan database. |
| **uuId** | The UUID of the item. |
| **parentUuid** | The UUID of the parent object. For photos from work orders this will equal **key1**, for survey photos this will equal the UUID of the survey instance they were taken on. |
| **key1** | The project ID as it is stored in the Pruvan database. |
| **key2** | The project ID visible to the user. |
| **key3** | The task ID as it is stored in the Pruvan database. |
| **key4** | The task name visible to the user. |
| **key5** | (unused) |
| **attribute1** | address1 from the project |
| **attribute2** | address2 from the project |
| **attribute3** | city from the project |
| **attribute4** | state from the project |
| **attribute5** | zip from the project |
| **attribute6** | country from the project |
| **fileExt** | The file extension of the photo, CSR or survey results. If the fileType = "picture" or "csr" then the fileExt is "jpg". If the fileType = "survey" then the fileExt is "pdf". If the fileType="check-in" then the fileExt is "json". If the fileType is "file" then the fileExt will be approprieate (ie pdf for a pdf file). |
| **fileName** | The high-level name of the file (presented to customers in the UI, Downloader, and Zip download). This is generated from a millisecond Unix epoch timestamp. It is generally a unique value within a given project, though not guaranteed. |
| **fileType** | The fileType of the file. If the fileType = "picture", then the file is a photo. If the fileType = "csr", then the file is a Certified Service Record. If the fileType = "survey", then the file is the results of a survey. If the fileType="check-in" then the file is plain text containing a JSON object. If the filetype is file then the file will be a file with an appropriate extension; current pdf is the only accepted file type for "file". |
| **evidenceType** | The evidence type to which the photo is associated. Typical values include "before", "during" and "after." These can be altered using Mobile Configuration Options. If your system does not support user-defined evidence types then you need to make sure to declare them as project or service level options in your projects. Certain photos taken on survey questions will have the evidenceType of that question's ID. Note that moving photos around in a survey does not change their evidenceType from the original value. |
| **survey** | The survey answer payload (JSON string) for surveys. (See Surveys section) |

| | |
|---|---|
| **template** | The survey template ID, can be prefixed by “{master_username}::”. |
| **notes** | Notes added to the photo from the device in the field. |
| **gpsAccuracy** | The radius of uncertainty for the location, measured in meters. |
| **gpsLatitude** | The latitude where the photo, CSR or survey was completed. |
| **gpsLongitude** | The longitude where the photo, CSR or survey was completed. |
| **gpsTimestamp** | The timestamp given by the GPS receiver. |
| **authenticated** | Pictures are authenticated if the security key generated at photo creation matches the security key of the photo at the time of upload. If the fileType = "picture", then the photo contents are authenticated if value is "1", not authenticated if value is "0". If fileType = "csr", then it’s a count of the number of authenticated photos for the CSR. |
| **locationDifference** | The difference of the GPS coordinates of the project address and the actual GPS coordinates of the device when the photo, CSR or survey was completed; measured in feet. |
| **csrCertifiedTime** | If the fileType = "picture", then the photo time is certified if value is "1", not certified if value is "0". If fileType = "csr", then it’s a count of the number of certified time photos for the CSR. |
| **csrLocationSource** | If the value is "mobile" or "Mobile2" then the photo was taken with the Pruvan app. |
| **csrPictureCount** | If the fileType = "picture", then the value is 1. If fileType = "csr", then it’s a count of the number of total photos for the CSR. |
| **csrTimeStampSource** | If the value is "mobile" or "Mobile2" then the photo was taken with the Pruvan app. |
| **csrCertifiedLocation** | If the fileType = "picture", then the photo location is certified if value is "1", not certified if value is "null". If fileType = "csr", then it’s a count of the number of certified location photos for the CSR. |
| **attribute7-15** | Work Order level **attributes7-15** |
| **attribute16-30** | Service level **attribute1-15** |
| **deviceId** | Device Id for the mobile device. iOS devices are prepended with "ios-" before the device id. |
| **phoneNumber** | The phone number for the device. |
| **status** | The status of the photo in the Pruvan system. |
| **uploaderVersion** | The version number of the Pruvan mobile app. |
| **batchId** | Upload batch id for the file |
| **workDay** | The work day date the file was created. |
| **timestamp** | The current time in UNIX timestamp (UTC). |
| **clientCode** | The clientCode for the project for which the file is associated. |
| **createdBy** | The master user |
| **createdBySubUser** | The subuser that took the item |
| **lastUpdatedBy** | The user who last updated this record. |
| **creationDate** | The date and time the file was created |
| **videoUri** | A link to the mp4 video file, for video submissions, valid for 24 hrs |

# Pruvan Status Management

Pruvan has a sophisticated status management system that allows you to keep track of your Pruvan Work Orders. The Pruvan Status Management System allows you to set clientStatus, status and track fieldStatus for each work order.

| Status | Description | Notes |
|---|---|---|
| **clientStatus** | Your work order status | This is set by you and represents your work order status in your system. It allows you to communicate the work order status to your Pruvan users. |
| **status** | The Pruvan Office status | This is set by Pruvan and allows you to track the current status of the work order in Pruvan's back office admin webUI. |
| **fieldStatus** | The Pruvan Field Status | This is set by Pruvan Mobile and tracks the status of the work order in the field. |

## Pruvan Status Values

Pruvan Status Values are visible in the Pruvan webUI and in the Pruvan Mobile App. The Pruvan Status communicates the status of the order in your office.

| Status | Description |
|---|---|
| **unassigned** | The work order is new and has not been assigned to a field worker. This is the initial status of an order |
| **assigned** | The work order has been assigned to a field worker |
| **accepted** | The work order has been accepted by the vendor |
| **rejected** | The work order has been rejected by the vendor |
| **rush** | A rush has been requested on the order |
| **rework** | The work order has been denied by quality assurance and rework is required |
| **canceled** | The work order has been canceled |
| **complete** | The work order has been completed in the field and is ready for quality assurance review |
| **invoiced** | An invoice for the work order has been submitted to the client |
| **submitted** | The work order has been submitted to the client for quality assurance review |
| **closed** | The work order has been closed |
| **deleted** | A special status that integrations can use to delete the work order |

## Pruvan Field Status Values

Pruvan Field Status values are visible in Pruvan Mobile and the Pruvan webUI. The Pruvan Mobile status communicates the status of the order in the field.

| Field Status | Description |
|---|---|
| **new** | New order, not viewed |
| **updated** | The order has been updated |
| **viewed** | Order has been viewed, not in process |
| **in-process** | A work item has been executed on this order (picture taken, survey submitted, etc.) |
| **field-accepted** | The order has been accepted by the field worker |
| **field-rejected** | The order has been rejected by the field worker |
| **complete** | The order has been marked field-complete by the field worker |

## Managing Pruvan Status

You can manage the Pruvan Status fields by setting the appropriate status field in your integration. Statuses are dependent on which integration type you decide to implement, to learn more about integration types (see the "Pruvan Integration Types" section below). Both internal and external integrations can set **clientStatus**. Only internal integrations can set the **status** field directly. External integrations can set the **status** indirectly if the integration's **auto_admin** option is set. Only Pruvan Mobile can set fieldStatus.

| Field | Internal Integration | External Integration | Pruvan Office (webUI) | Pruvan Mobile |
|---|---|---|---|---|
| **clientStatus** | Y | Y | N | N |
| **status** | Y | N | Y | N |
| **fieldStatus** | N | N | N | Y |

## Setting the Pruvan Status

Internal integrations have direct insert/update access to **clientStatus** and **status** so both should be set.

External integrations have direct access to **clientStatus** and indirect access to **status** through the auto_admin integration option. If auto_admin is set the Pruvan external integration will allow **status**, **dueDate** and **instructions** to be set by the external integration. To support the internal integration's auto_admin mode you must map your system's status (the **clientStatus** in Pruvan) to a Pruvan status.

For example if your system's status for requesting an order expedite is "Expedite" you should set **clientStatus** to 'Expedite' and set the **status** field to the Pruvan status of 'rush'.

## The status Web Service

The status web service call is initiated from Pruvan anytime the status of your work order in Pruvan changes. This allows you to keep track of status changes on your work orders that have been interfaced to Pruvan. Anytime the status changes on one of your work orders - you will receive a status update via the status web service. Note that no status message will be sent when the status of a work order is set to "Unassigned" to prevent meaningless status updates when vendors are performing re-assignments.

### status web service call

The status web service will receive the JSON payload below when the status of your work orders in Pruvan change. It will be sent to {web service url}/status.{web service extension} in the POST variable "payload":

```
{
  "username": "someuser",
  "password": "d28069c84398483874d2cba15f10ffdd996ec6f5",
  "token": "dda25c60-4fce-11e4-916c-0800200c9a66",
  "workOrders": [
    {
      "workOrderNumber": "SMC::Simple",
      "clientStatus": "assigned",
      "status": "assigned",
      "fieldStatus": "viewed",
      "dueDate": 1414434590
    }
  ]
}
```

### Using the status web service

The status web service call is useful for keeping track of status changes of your orders in Pruvan. You will receive a status update when a work order is completed for example.

You can use the information to update the history of your work order or build sophisticated work flows.

If the status is processed successfully you should return the following as a JSON string:

```
{"status":true,"error":null}
```

If there's an exception, you should return the following JSON object with an array of workOrderNumber, error records for each status that failed:

```
{"status":false,
 "error": [{
                          "workOrderNumber":  “Simple1”,
                          "error": "work order does not exist"
             }
  ]
}
```

# Pruvan Integration Types

There are two types of integrations in Pruvan – Internal and External (formerly Mode 1 and Mode 2 respectively). An Internal integration is an integration that is designed to extend a host organization's current work order management system with Pruvan Mobile. An External Integration is an integration that allows Pruvan users to import your work orders into their Pruvan accounts and manage them alongside work orders from other work providers.

## Choosing an Integration Type

An Internal (formerly Mode 1) integration is an integration that is designed to extend an organization's current work order management system with Pruvan Mobile. A Host organization will implement an internal integration if they primarily consider the field work force "users" of their system and they have less than 5000 active work orders/day.

An External (formerly Mode 2) integration is an integration that is designed to provide your vendors the ability to use Pruvan as a Mobile option for managing and completing work for you. You would implement an External Integration if you primarily consider your field work force "vendors" and you have more than 5000 active work orders/day.

| Integration Features | Internal Integrations | External Integrations |
|---|---|---|
| **Host organization pays for its users and the user is provided with a Pruvan login (no signup required)** | Y | N |
| **Host organization's vendors sign up for Pruvan and pay for their own usage (vendor Pruvan Account required)** | N | Y |
| **Host organization's order volume exceeds 5000 active work order/day** | N | Y |
| **Host organization's staff users can perform QA in Pruvan (all photos are available to staff in Pruvan)** | Y | N |
| **Host organization's vendors can perform QA in Pruvan** | N | Y |

## Internal Integrations (formerly Mode 1)

Internal integrations can directly set **assignedTo, status, dueDate**, **instructions and description** as it is assumed the field work force are users of the host organization.

## External Integrations (formerly Mode 2)

External integration cannot directly set **assignedTo, status, dueDate**, **instructions and description** as the field work force is a vendor of the host organization and needs to manage their field workers themselves. A small vendor that does not have a back office admin operation can set the auto_admin option to allow the external integration to set these fields.

## Pruvan Share

Pruvan provides a powerful feature called Pruvan Share that allows an Internal Integration to be accessed like an External Integration giving your users the best of both worlds. This allows you to deploy an Internal Integration and take advantage of the ability to pay for your users (and not require them to sign up for their own Pruvan account) – but also allow them to get their own Pruvan account when they grow to a vendor with multiple crews. Pruvan Share allows them to access their user account in your Internal Integration as a Pruvan Share Integration. This pulls all their work from your Integration directly into their own Pruvan account. You can read more about Pruvan Share here.

Pruvan Share is a good option for host organizations with less than 5000 active work orders/day. Host organizations with a higher order volumes should implement External Interfaces to ensure good performance and avoid fees for exceeding the 5000 active orders/day threshold.

## Pruvan Configuration Options

Configuration Options allow you to customize the way Pruvan Direct and Pruvan Mobile operate. Options can be set at the Device, User, WorkOrder, and Service levels. The Pruvan API allows you to set options on your work orders and services.

Options are added to a workOrder or service object in the options element.

Example:

```
{
  "workOrders": [
    {
      "workOrderNumber": "Simple",
      "address1": "110 East Main Street",
      "city": "Round Rock",
      "state": "TX",
      "zip": "78664",
      "services": [
        {
          "serviceName": "Task",
          "options": {
            "camera_orientation": "portrait"
          }
        }
      ],
      "options": {
        "camera_orientation": "landscape"
      }
    }
  ]
}
```

The following options are supported:

| Option Name | Device | User | WorkOrder | Service |
|---|---|---|---|---|
| all_required_tasks_complete | | x | x | |
| allow_manual_work_order_creation | | x | | |
| auto_relay | x | x | x | x |
| camera_icon_hidden | | x | x | |

| | | | | |
|---|---|---|---|---|
| **camera_orientation** | x | x | x | x |
| **complete_icon_hidden** | | x | x | |
| **evidenceTypes** | | x | x | x |
| **mark_complete_label** | | x | x | |
| **mark_complete_mode** | | x | x | |
| **photo_size** | x | x | x | x |
| **photo_timestamp** | x | x | x | x |
| **required** | | | | x |
| **upload_pics_only_when_wo_complete** | x | x | x | |
| **frame_images** | | x | x | |
| **check_in_required** | | | x | |
| **check_in_provider** | | | x | |
| **check_in_photo_required** | | | x | |
| **check_in_relay** | | | x | |

Please see the Pruvan Options documentation (https://pruvan.zendesk.com/hc/en-us/articles/206678913-Mobile-Configuration-Options) for the latest details on supported options and their values.

# Pruvan Push API

The Pruvan Push API allows you to keep your system synchronized with Pruvan in real-time. The Push API is enabled by the Pruvan pushkey. The pushkey is generated when a Pruvan user defines an integration to you. The push key can be used to push order updates from your system to Pruvan for a specific user.

## Retrieving the pushkey

Pruvan sends a user's pushkey to your validate web service when the user clicks "Validate" in the Pruvan Create/Edit Integrations form. We suggest that you code your validate web service to grab the pushkey and save it in your user table.

## Using the pushkey

The Pruvan pushkey is a pre-authenticated url that you can post Pruvan workOrder updates to. When you create/update or delete orders in your system you can call the pushkey to update those specific orders in Pruvan. The call is simply a post to the pushkey's url.

Here is an example of a Pruvan pushkey:

```
https://direct.pruvan.com/api/?apiKey=Qc71lby4C_cQUco_YP71ey1RV7YelYNnsKqUJSgmZv1LgUlfqjB1PvTysS
qUnPUQw-fdHW9K-bm8egMCr-RmhvM6CrhqB9aRFKlglk1xi6UzvdFcWZDaVp4Tl3U_adsmmFcOYqEhf5k
```

To update the status of workOrder "Simple" to "rush" you would post to the specific user's pushkey with the following JSON payload in a POST parameter of "workOrders" with a content-type of "application/x-www-form-urlencoded":

```
workOrders= {
  "workOrders": [
  {
    "workOrderNumber": "Simple",
    "clientStatus": "rush"
  }
 ]
}
```

The Pruvan Push API will respond with:

```
{"response":{"created":0,"updated":1,"deleted":0},"error":null}
```

Multiple orders are supported and errors are returned in an array with the workOrderNumber and error. This allows you to synchronously update Pruvan as your system creates, updates and deletes orders. The Push API can also be used to bulk load orders that exceed the orders/call limit of getWorkOrders.

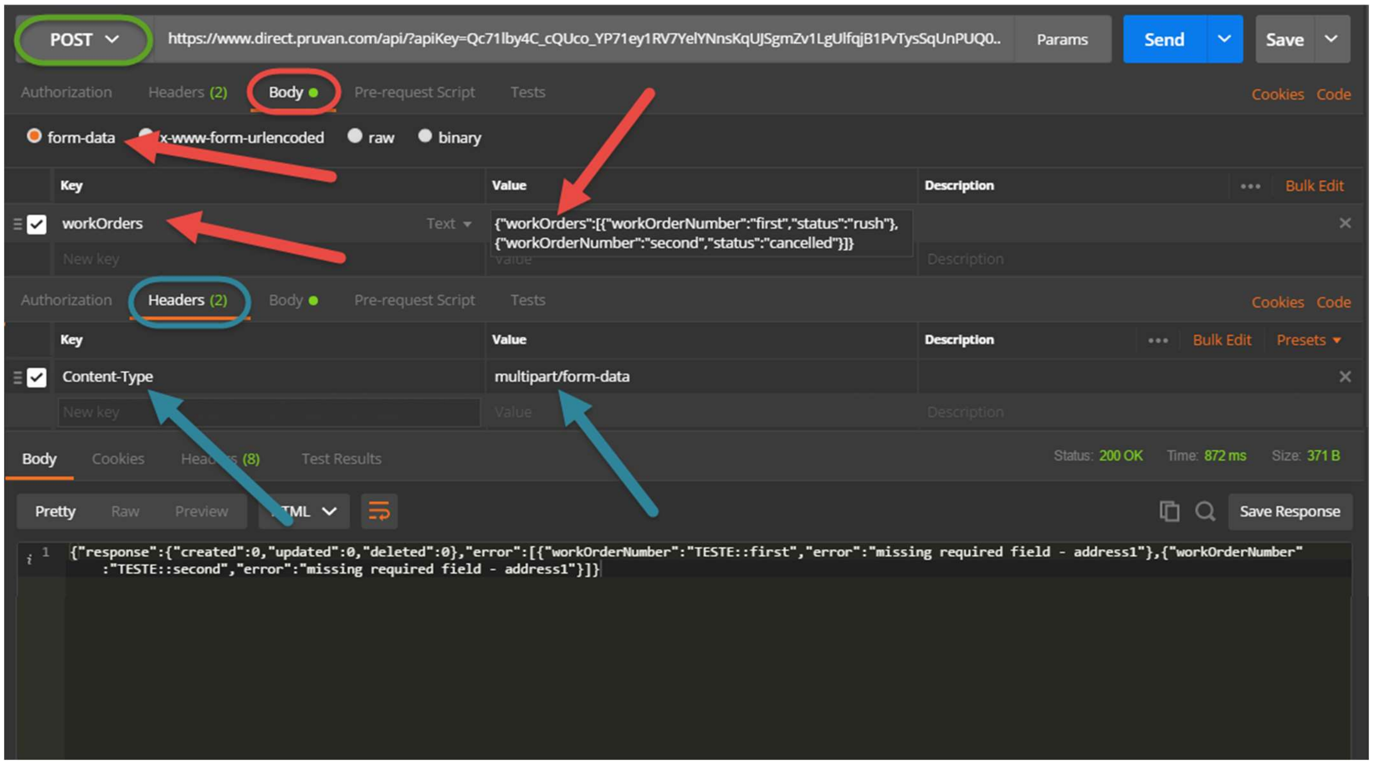
## Force Updates

Updates can be made to orders created on external integrations that typically do not allow updates. For instance you can force update the assignedTo on a project while normally an update on assignedTo is ignored. To use Force Update add the ‘force’: true key-value pair to the each work order you want to force update:

```
workOrders= {
  "workOrders": [
  {
    "workOrderNumber": "Simple",
    "status": "assigned",
    "assignedTo": "vendor1",
    "force": true
  }
 ]
}
```

An example using Postman to push orders in to Pruvan:

- A POST to your integration client’s Push Key URL – Green
- A Content-Type: multipart/form-data Header – Blue
- A body set to Form-Data, with the key (field name) *workOrders*, and the value set to the JSON object containing the workOrders array – Red
- A response from Pruvan with the Header Content-Type: text/html and a body containing a JSON object representing the number of orders created, updated, and deleted



POST https://www.direct.pruvan.com/api/?apiKey=Qc71lby4C_cQUco_YP71ey1RV7YelYNnsKqUJSgmZv1LgUlfqjB1PvTysSqUnPUQ0..

Body: form-data

| Key | Value | Description |
|---|---|---|
| workOrders | {"workOrders":[{"workOrderNumber":"first","status":"rush"},{"workOrderNumber":"second","status":"cancelled"}]} | |

Headers (2)

| Key | Value | Description |
|---|---|---|
| Content-Type | multipart/form-data | |

Status: 200 OK Time: 872 ms Size: 371 B

```
{"response":{"created":0,"updated":0,"deleted":0},"error":[{"workOrderNumber":"TESTE::first","error":"missing required field - address1"},{"workOrderNumber"
   :"TESTE::second","error":"missing required field - address1"}]}
```

## Pruvan Surveys (Mobile Forms)

Pruvan's Survey/Mobile Forms engine allows you to design custom surveys and mobile forms for your users and get the results back in both JSON and .pdf formats. To use Pruvan Surveys through the Pruvan API you simply need to create survey templates in your Pruvan Account and then reference those surveys in the **survey** field of a workOrder/service object. You can also reference public surveys owned by Pruvan or other Pruvan users. Please see the Pruvan Surveys documentation below for more details on setting up and using surveys:

- Pruvan Surveys – Quick Start Guide
- How to use the Pruvan Survey Builder
- Best Practices Guild for building Pruvan Surveys

The uploadPictures web service call is initiated from Pruvan when photos, surveys or checkins are published. The uploadPictures web service is a "push" request from Pruvan to your system.

This request will contain metadata for photos, surveys and checkins. The uploadPictures JSON object will be the same basic structure for all file types. Surveys and checkins have additional elements for survey and checkin details respectively (Please see the uploadPictures Web Service section for more details).

For survey uploads you will receive a **survey** field that contains a JSON object that defines the completed survey.

## Example Survey

```
{
  "meta": {
    "currentQuestionIndex": 1,
    "surveyId": "2602cfc4-e6e5-4a8a-ae31-cca90aa2f6a2",
    "parentId": "148",
    "surveyTemplateId": "vendor1::Simple-v1"
  },
  "answers": [
    {
      "answer": [
        "blue"
      ],
      "id": "question1",
      "picIds": [
        "a52eafbf-cfd2-45f6-8351-2cff2bec1d3b"
      ],
      "picFileNames": [
        "1415913831993.jpg"
      ],
      "question": "What is your favorite color?  ",
      "answerType": "chooseOne",
      "answerOptions": [
        {
          "answer": "red",
          "display": "Red"
        },
        {
          "answer": "blue",
          "display": "Blue"
        }
      ]
    }
  ]
}
```

## The survey Object

| Element | Description |
|---|---|
| **answers** | An array containing an object for each question in the survey |
| **answers->answer** | The answer to this question |
| **answers->id** | The questionId for this question as defined in the template |
| **answers->question** | The text of the question for this question |
| **answers->picFileNames** | An array of picture fileNames that were taken for this question |
| **answers->picIds** | An array of pictureIds that were taken for this question |
| **answers->answerType** | The type of this question as defined in the survey template for this question |
| **answers->answerOptions** | The acceptable answers for this question as defined in the template |
| **meta** | An array containing metadata for this survey |
| **meta->surveyId** | The uuid of this survey |
| **meta->parentId** | The parent workOrderId of this survey |
| **meta->surveyTemplateId** | The survey template used for this survey |

The answers array will contain an answer object for each question answered in the survey.

Pruvan also sends a standard survey report in .pdf format as multipart/form-data in the **file** field. You can store this .pdf file on your system and use it for any purpose.

For more information on Pruvan Surveys see:

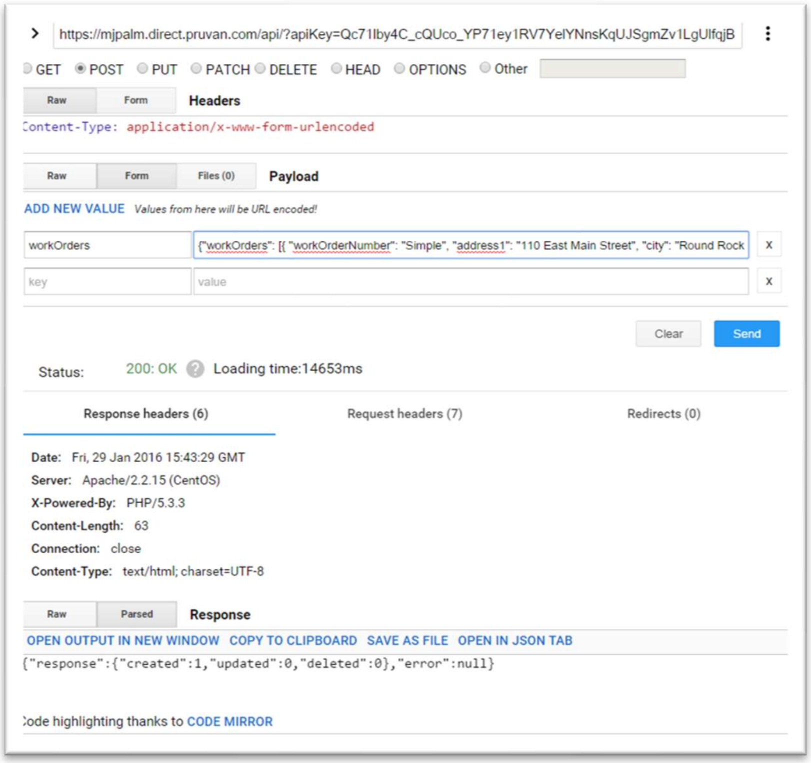
- Pruvan Surveys – Quick Start Guide
- How to use the Pruvan Survey Builder
- Best Practices Guild for building Pruvan Surveys

## Testing

You can test the Pruvan Push APIv2 by posting a workOrders payload to your pushkey url. You can retrieve your pushkey url by editing your integration definition in the Integrations form. You can create sample workOrders payload by simply editing a JSON file.

```
{
	"workOrders": [{
		"workOrderNumber": "Simple",
		"address1": "110 East Main Street",
		"city": "Round Rock",
		"state": "TX",
		"zip": "78664",
		"services": [{
			"serviceName": "Task"
		}]
	}]
}
```

Using a REST testing tool like the “Advanced Rest Client” (https://code.google.com/p/chrome-rest-client/) Chrome plugin you can send the payload to the Pruvan Push APIv2 and examine the results. The above example will create a work order “Simple” with one service. Create a form field named workOrders and set the value to the above JSON. The method must be “POST” and content-type must be “application/x-www-form-urlencoded”.



https://mjpalm.direct.pruvan.com/api/?apiKey=Qc71lby4C_cQUco_YP71ey1RV7YelYNnsKqUJSgmZv1LgUlfqjB

GET [✓ POST] PUT PATCH DELETE HEAD OPTIONS Other

Headers

```
Content-Type: application/x-www-form-urlencoded
```

Payload

ADD NEW VALUE *Values from here will be URL encoded!*

| | |
|---|---|
| workOrders | {"workOrders": [{ "workOrderNumber": "Simple", "address1": "110 East Main Street", "city": "Round Rock |
| key | value |

Status: 200: OK Loading time:14653ms

Response headers (6) Request headers (7) Redirects (0)

```
Date: Fri, 29 Jan 2016 15:43:29 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.3.3
Content-Length: 63
Connection: close
Content-Type: text/html; charset=UTF-8
```

Response

```
{"response":{"created":1,"updated":0,"deleted":0},"error":null}
```

Pruvan also provides a useful web services testing tool that you can test your web service endpoints with. You can access it here: https://www.direct.pruvan.com/admin/api-validate. This tool will test your Pruvan integration endpoints by sending validate, getWorkOrders and uploadPicture payloads and testing their response.

# General

All strings sent should only contain ASCII characters. UNICODE strings and/or characters are not accepted.

All errors should be returned in valid JSON format in the appropriate error element, containing a message that is appropriate for an end user. Error messages will be displayed as-is in the Pruvan UI.

Free Standard Driver Integrations are limited to 5000 inserts/updates per hour. Named drivers are available for purchase and have higher import thresholds available; contact Pruvan Sales for more information.

## Date Fields

UNIX timestamps are accepted as well as fully qualified date strings. When sending a date string you should include the time zone in number offset format, that is "-06" for CST or "+00" for UTC. For any date fields that you do not supply a time, the system will assume midnight Central Time (CST / CDT).

### dueDate

Specifically for the dueDate field there is a special consideration. We will alter the time on import so that the same date is displayed in most of the world. This is done by taking your timestamp, converting it to UTC, and setting the time to 1200hrs. Therefor you need to ensure that the date you want displayed, in the dueDate field in Pruvan, is the date you want after conversion to UTC. The easiest way to do this is to just send the date in UTC already. For example; if you want "5/25/2016" displayed, then send "2016-05-25 1200 +0". This is to avoid potential date-shifting pitfalls. For instance if you sent "2016-05-25 2359 -07", that would convert to "2016-05-26 0659 +0", and then we would set the time to noon "2016-05-26 1200 +0", and you would have 5/26 displayed everywhere as opposed to 5/25.