SecureSync, VersaSync, and NetClock

Technical Note

Orolia REST API Developer Guide

This technical note applies to the following timing products:

- SecureSync® 1200 model
- SecureSync® 2400 model
- NetClock® 9400 series
- VersaSync®

Overview

The graphical Web User Interface ("Web UI") used with Orolia's time servers has a built-in REST API which allows for status and configuration data to be sent and retrieved from the device without having to use the Web UI. This is useful for machine-to-machine communication, as well as for developing mobile apps. The REST API uses the JSON data format when performing HTTP GET and POST operations.

To perform these HTTP operations, it is necessary to know the data format required to retrieve and send data to a timing server. Orolia's **Postman**™[1] collection is a full documentation of available endpoints within the Web UI, with examples of how to pull and send data through the API, and saved responses in order to visualize the requests.

About Postman

Postman is an HTTP client that serves as a development app to prototype and test APIs. The Postman app is available for Microsoft Windows™, Mac OS X or later, and Linux; there is also a web browser version: http://www.postman.com/downloads/.

[1]Postman is a software app developed by Postdot Technologies, Inc. In its basic configuration, the tool is free of charge.

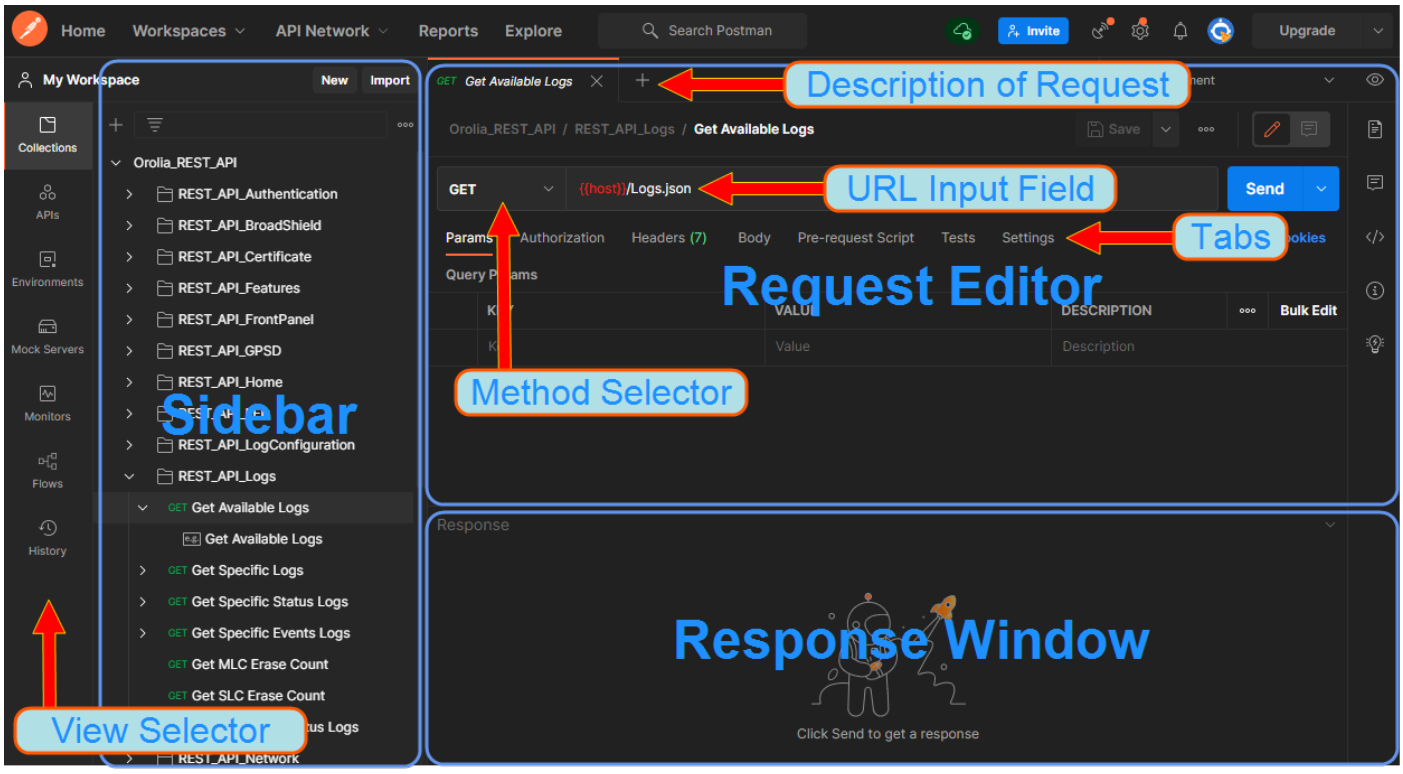
Postman can be used to send the requests to a Time Server unit, and it will return the JSON response from the device. This allows the user to quickly test API calls without having to develop test software, and the format of the data returned can be easily analyzed for inclusion into scripts or applications that can consume the data.

Downloading and Installing Postman

1. Navigate to http://www.postman.com/downloads/, and select "Download the App" or "Try the Web Version".
2. Install and launch Postman.
3. Create an account by signing up. This will ensure your requests, collections, environments and history data are saved for future reference.

Familiarizing Yourself with Postman

The following is a brief overview of the Postman UI. More comprehensive assistance can be found under https://learning.postman.com/docs/getting-started/introduction/.



The **View Selector** is used to switch between different functionality.

The **Sidebar** lists the requests stored in the loaded Collections tab (see "Orolia REST API Developer Guide" on page 1), the parameters stored in the Environments tab, and – under the History tab – a list of recently submitted requests.

The Request Editor is used to configure elements of a request. For example:

- **URL**: Type the address of the endpoint that you want to call into the URL input field. URLs used previously will be suggested via auto-complete.
  Note that parameters entered in the URL input field bar or in the key/value editor will not automatically be URL-encoded. To manually encode the parameter value, right click on a highlighted text, and select EncodeURIComponent .
- **Method**: Select the http operation that you want to use (GET, POST, etc.).
- **Tabs**: Use the tabs in the Request Editor to configure the requests. For example:
  - **Headers**: Click on the **Headers** tab to display the Headers key-value editor. The header frequently contains fields for authentication, Cookies, a time stamp, MD5 sums, and MIME-content type information.
  - **Body**: Click the **Body** toggle switch to open the Body editor. The body editor has different controls depending on the body type: **form-data**, **urlencoded**, **raw**, **binary**, and **GraphQL**.
  - To edit key values, click the **Params** button.

Postman functionality highlights:

- Create requests by conveniently specifying Method, URL parameters, Header and Body.
- Submit API calls quickly to test scripts; generate code snippets that can be copied and pasted.
- Specify authorization to be used.
- Display responses in different formats e.g., "pretty", "raw", or as rendered HTML pages.
- Organize and store requests in **Collections**.
- Store request parameters that will be used repeatedly (e.g., keys and values used as login credentials) in development project-specific **Environments**.
- Access history of sent requests.
- Capture documentation for requests in a description field.

Importing the Orolia Collection

Orolia's **Postman**™ collection for Time Servers provides examples of how to pull and send data through the API.

To import this collection:

1. Unzip the Orolia REST API kit to a local directory of your choice.

   The Orolia Rest API kit includes the following files:

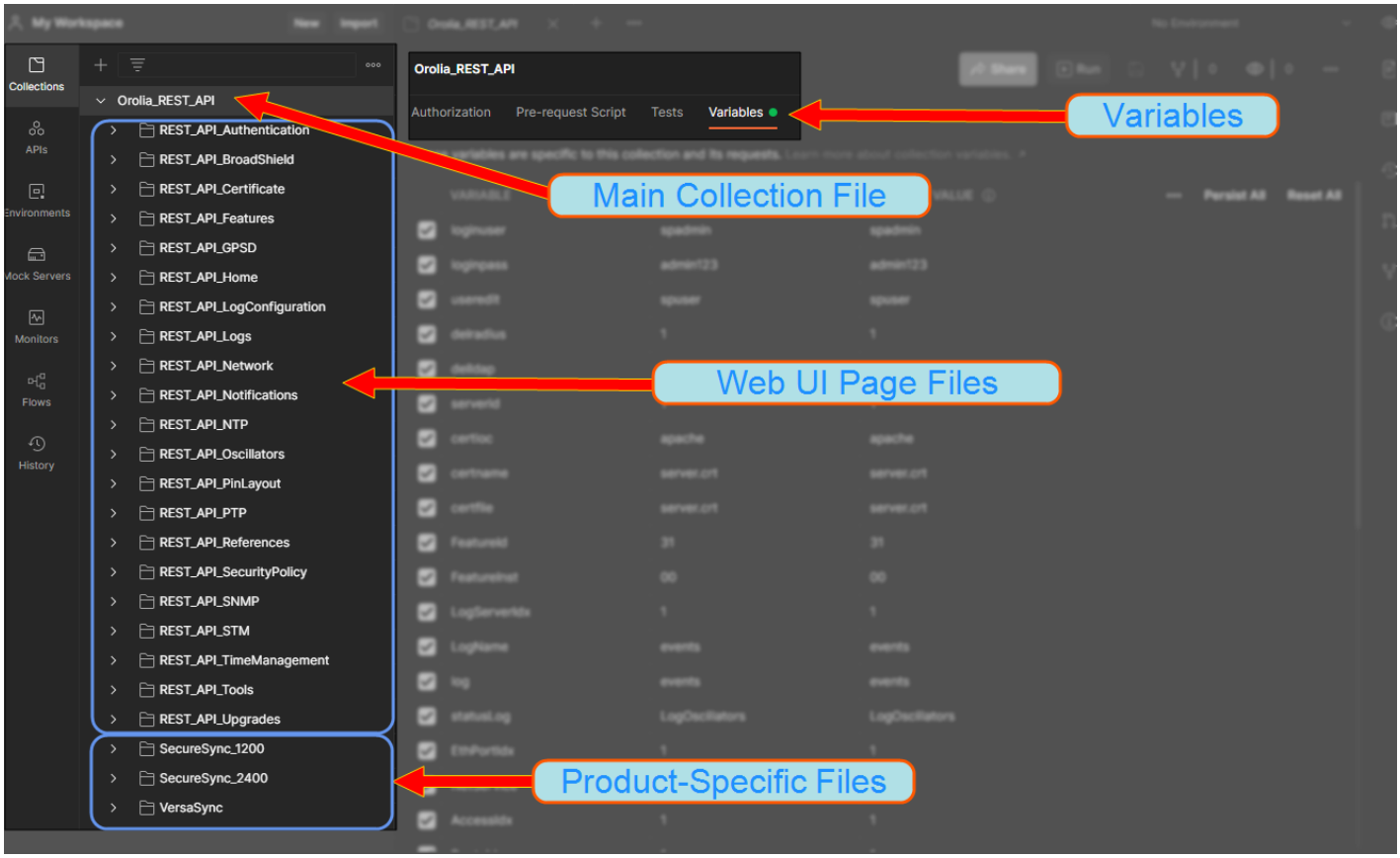
   > `Orolia_REST_API_vX.postman_collection.json`

   > `OroliaREST_APIdevelopersGuide.pdf` (this document)

2. Open the Postman app, using the credentials of your previously created account.
3. Import the Orolia REST API Collection:
   a. For the standalone app, select **Import**.
   b. Drag and drop or click **Upload Files** and select the `Orolia_REST_API_vX.postman_collection.json` file from your previously chosen upzip location.
   c. Under the Collections tab in the Sidebar on the left, Orolia_REST_API will be displayed. Click on it to display the Collection's folders which reflects the menu structure of the SecureSync Web UI e.g., Networking, Log Configuration, NTP, etc. Each folder contains requests. Click on any request to display it in the Request Editor.
4. It is not necessary to import an Environment into Postman, because all of the necessary variables are attached to the collection itself.
   (Note: former versions of these files, as well as some product-specific collections, will need an Environment file. If this is the case, one will be included with your REST API documentation. It should be imported into Postman in the same manner as the collection).

About the Orolia REST API Collection

The Orolia REST API Collection will appear as follows after importing into Postman:



Variables

The Orolia REST API collection has all the necessary information included as **Variables** attached to the Main Collection File (these variables can be viewed when the Main Collection File is selected in the sidebar). The Variables of the collection can be altered to fit the user's network and communication parameters.

Web UI Page Files

The Main Collection File is divided into sections based on the pages of the Web UI. They are organized alphabetically with the addition of the REST_API_ prefix. For instance, the SNMP functions can be found on the SNMP page of the Web UI, and are labeled as REST_API_SNMP in the collection.

Product-Specific Page Files

The REST API collection also includes product-specific files which include instructions and endpoints that are only functional for specific products. This is especially important for users with a **SecureSync 1200**, because many of

the endpoints listed in the collection do not apply to that model. If an endpoint is listed in the SecureSync 1200 file, the endpoint instruction should be used from that file, rather than from the data in the main collection.

Since the SecureSync 1200 is an older model, some of the endpoints do not follow the same format and/or syntax as described below. When that is the case, those endpoints are also documented in the product-specific collection.

About the Orolia REST API for the Time Server Web UI

Orolia's Web UI has been designed following the M-V-C (Model – View – Controller) pattern.

Each View correlates to a function in the Controller. The Controller will take the data, manipulate it (as needed), and send it to the Model. The Model is the back end of the Web UI: Here the logic is applied to the data , using the CRUD functionality: Create, Read, Update, Delete.

Whenever a user accesses a page, the View component issues a GET request to the Controller, since the user ultimately wants to retrieve (or: GET) the data that is displayed on the loaded page. If, however, a user wants to add or configure a setting, the View component will issue a SET (or: POST) request to the Controller. In both cases, the Controller will receive the request, decide which operation to apply (CRUD), and then forwards the processed request to the Model, which will execute the request.

Next to SecureSync and VersaSync's Web UI, the API offers an alternative means to communicate with the Controller: If e.g., a Login request is submitted to a time server unit by sending this request to the unit's URL, the http POST "Login" request will, in fact, call into the "Users" Controller to execute the Login operation.

Syntax Example of an API Request

The format of the above-mentioned POST "Login" call would be as follows:

» `{{host}}/users/login.json`

In this example the url points to the recipient, and users/login requests access to the "Login" function inside the "Users" controller. The json appendix specifies that the data format to be used is the JSON format. Other formats can be XML, CSV, DY (a proprietary Orolia format for graphs).

Technical Support

To request technical support for your time servers unit, please go to the"Timing Support" page of the Orolia website (https://www.orolia.com/support/timing), where you can not only submit a support request, but also find additional technical documentation.

You can also email us directly at timingsupport@orolia.com.

Phone support is available during regular office hours under the telephone numbers listed below.

To speed up the diagnosis of your time servers, please send us:

- the current **product configuration**, and
- the **events log**, if applicable.

Thank you for your cooperation.

Regional Contact

Orolia operates globally and has offices in several locations around the world. Our main offices are listed below:

Country	Location	Phone
France	Les Ulis	+33 (0)1 64 53 39 80
USA	West Henrietta, NY	+1 585 321 5800

Additional regional contact information can be found on the Contact page of the Orolia website (https://www.orolia.com/contact-us).

- end of document -