

oneloTa User Guide



Revision	Date	Comment
1.0	February 8, 2016	Published User Guide
1.1	April 24, 2016	Updated with OCF name; addition of Derived Models and other additional sections
1.2	June 11, 2019	Editorial updates
1.3	October 3, 2019	Updates to derived models

NOTHING CONTAINED IN THIS DOCUMENT SHALL BE DEEMED AS GRANTING YOU ANY KIND OF LICENSE IN ITS CONTENT, EITHER EXPRESSLY OR IMPLIEDLY, OR TO ANY INTELLECTUAL PROPERTY OWNED OR CONTROLLED BY ANY OF THE AUTHORS OR DEVELOPERS OF THIS DOCUMENT. THE INFORMATION CONTAINED HEREIN IS PROVIDED ON AN "AS IS" BASIS, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE AUTHORS AND DEVELOPERS OF THIS DOCUMENT HEREBY DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, STATUTORY OR AT COMMON LAW, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. OPEN CONNECTIVITY FOUNDATION, INC. FURTHER DISCLAIMS ANY AND ALL WARRANTIES OF NON-INFRINGEMENT, ACCURACY OR LACK OF VIRUSES.

The OCF logo is a trademark of Open Connectivity Foundation, Inc. in the United States or other countries. *Other names and brands may be claimed as the property of others.

Copyright © 2016, 2019 Open Connectivity Foundation, Inc. All rights reserved.

Copying or other form of reproduction and/or distribution of these works are strictly prohibited

Table of Contents

Introduction	3
oneiota Overview	3
User Registration	4
License Agreements.....	4
User Types.....	4
Data Model Status Types.....	5
Getting to Know the Site	5
All Models	5
Proposals	6
Entering Data Models:.....	7
Uploading Data Models:.....	8
Sharing Proposals:	9
Submitting Proposals:	9
Approval Process.....	11
Releases.....	11
Downloading and Editing Released Data Models:	12
Git Repository.....	12
Derived Models.....	14
Simple Mapping	14
Conversion.....	15
Additional Derived Model Examples	16
Derivative Model Validation	16
What's Next?.....	17

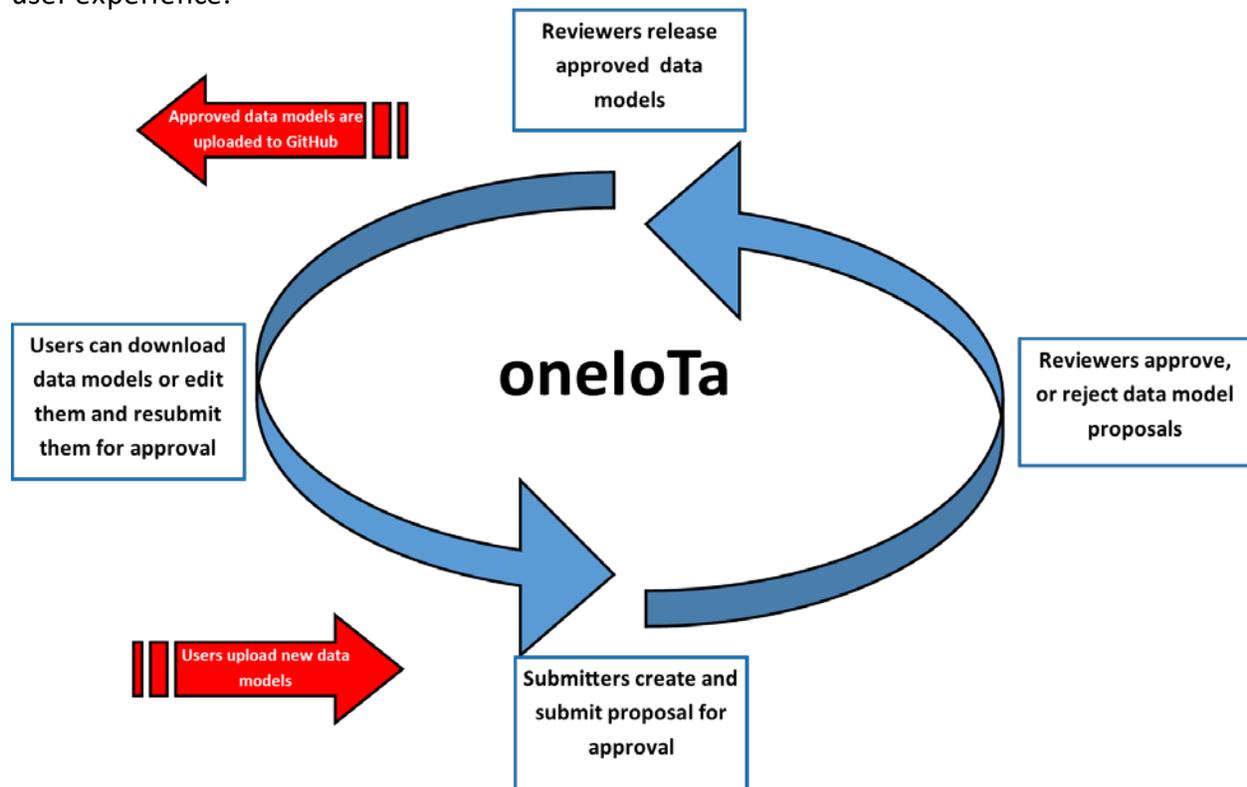
Contact Information:

Open Connectivity Foundation
3855 SW 153rd Drive
Beaverton, Oregon 97003
Phone: +1.503.619.0673
Email: oneiota@openconnectivity.org

If you have any questions, please contact oneiota@openconnectivity.org

Introduction

The Open Connectivity Foundation (OCF) was established with the goal of providing a common scalable standard for the Internet of Things with a certification tool that could ensure interoperability and an open-source implementation that could accelerate implementation time. In order to meet these objectives, a RESTful architecture was developed that would limit the system to just five (CRUDN) APIs and a constructive data model that would create complex devices and systems as collections of simpler devices and resources. It was also important to ensure interoperability with other ecosystems in order to expand the market and simplify the user experience.



oneloTa Overview

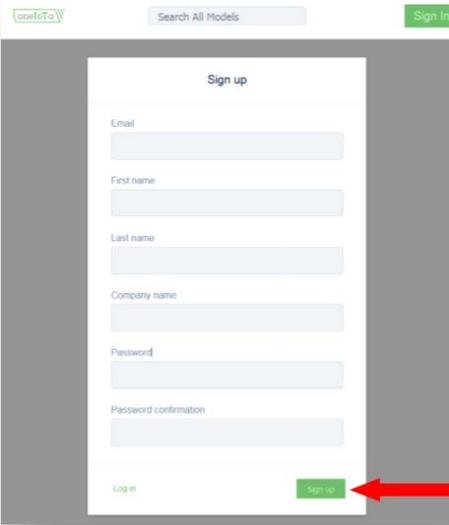
oneloTa (one Internet of Things architecture) is essentially an Integrated Development Environment (IDE) that sits at the center of these requirements and delivers on this promise using OCF. It has integrated, syntax-colored, and validating text editors for the needed OpenAPI Specification 2.0 (OAS 2.0 fka Swagger) schema that defines each model. It supports a crowd-sourced process that allows anyone to submit model proposals, and a back-end approval process that allows multiple organizations to approve models for their particular ecosystems. Finally, it's backed by a git repository so anyone can get the models without ever using the tool if they choose to do so.

If you have any questions, please contact oneiota@openconnectivity.org

User Registration

To register for a user account on the oneloTa site, go to http://oneiota-production.herokuapp.com/users/sign_up, enter your contact information and create a password. Your username is your email address. If you forget your password, click **Forgot your password?**

Once you click **Sign-Up**, you will be sent a verification email to the email address you entered. In the account verification email, there will be a link titled, “**Confirm my account,**” which you will need to click to authenticate your user account, and gain access to the site. If you do not receive the verification email, please check your spam folder.



License Agreements

All new users will automatically receive *Viewer* status (see user status definitions below). Before downloading a released data model, you must accept the **License Agreement**. Before a user can become a *Submitter* and contribute new data models, you must accept the **Contributor Agreement**. A notification will then be sent to oneloTa admin, who will verify your status as a *Submitter*.

User Types

There are four types of users supported in the oneloTa tool. User rights are cumulative. The roles are listed here from most restrictive to least restrictive.

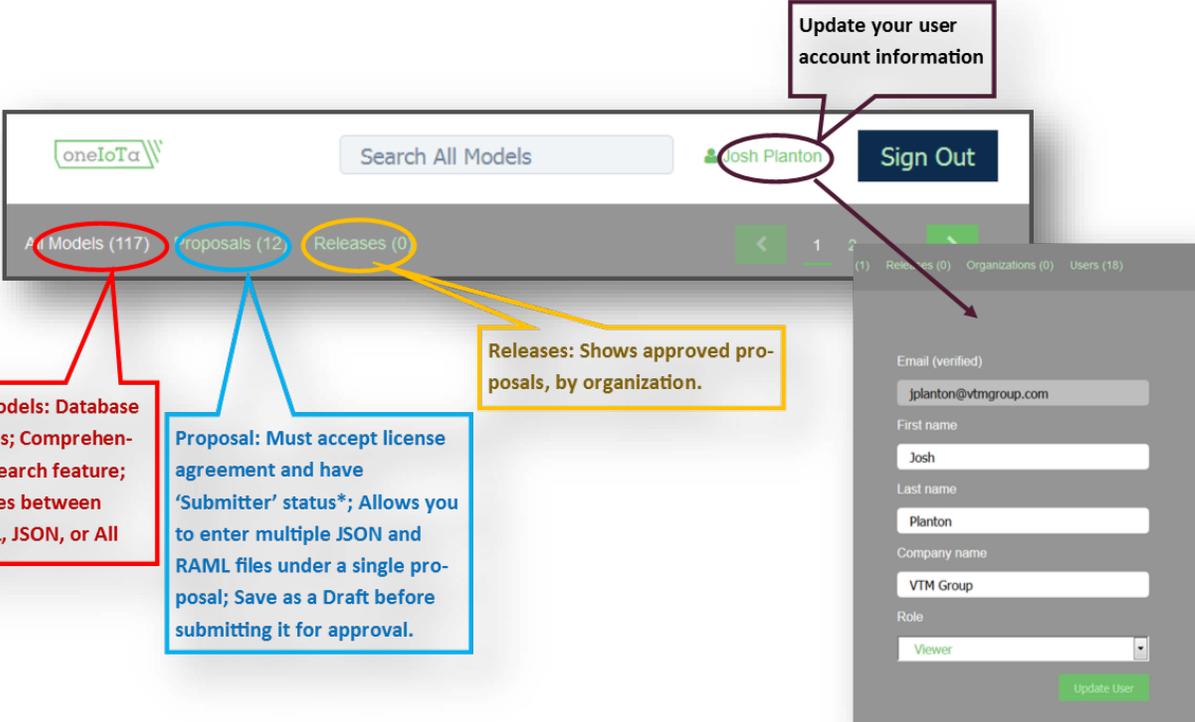
- **Viewers** – Have the ability to view and download approved and pending data models. Included files are discovered and linkable, so the device model tree can be easily traversed. The models list can be filtered by approval status and organization with the filter drop-down at the top right of the models screen.
- **Submitters** – Have agreed to the Contributors Agreement and are able to create and submit new data model proposals to OCF, or other approved organizations for their consideration. Submitters are able to save drafts of their proposal and return to them later.
- **Reviewers** – Reviewers, a.k.a “Managers,” of their respective organization, have the authority to approve or reject submitted proposals based on the organization’s processes, outlined within their internal procedures. Once a proposal is submitted, a reviewer can change the status of a model from “pending” to “approved.” A reviewer can also tag a release. Typically, there will only be a handful of reviewers for a particular organization and they will determine their own process for reviewing models and tagging releases.
- **Admin** – Has authority to approve new user status and review proposals and all data models.

Data Model Status Types

- **Approved** – device definition is approved and will be added to the list of official devices supported by the organization
- **Pending** – device definition is of interest, but has not yet been approved, still has errors or omissions, or is otherwise not ready for approval
- **Rejected** – device definition is duplicative, non-compliant, or otherwise not of interest to the organization

Getting to Know the Site

Once you login to the site you will see three main tabs across the header: *All Models*, *Proposals*, and *Releases*. In the upper right corner you will also find a link to your user profile, which will allow you to edit your user profile, and view your current user status.



The screenshot shows the user interface for a user named Josh Planton. The header includes the oneIoTa logo, a search bar for "All Models", and a "Sign Out" button. Below the header are three main navigation tabs: "All Models (117)", "Proposals (12)", and "Releases (0)". A user profile dropdown menu is open, showing fields for email, first name, last name, company name, and role, with an "Update User" button.

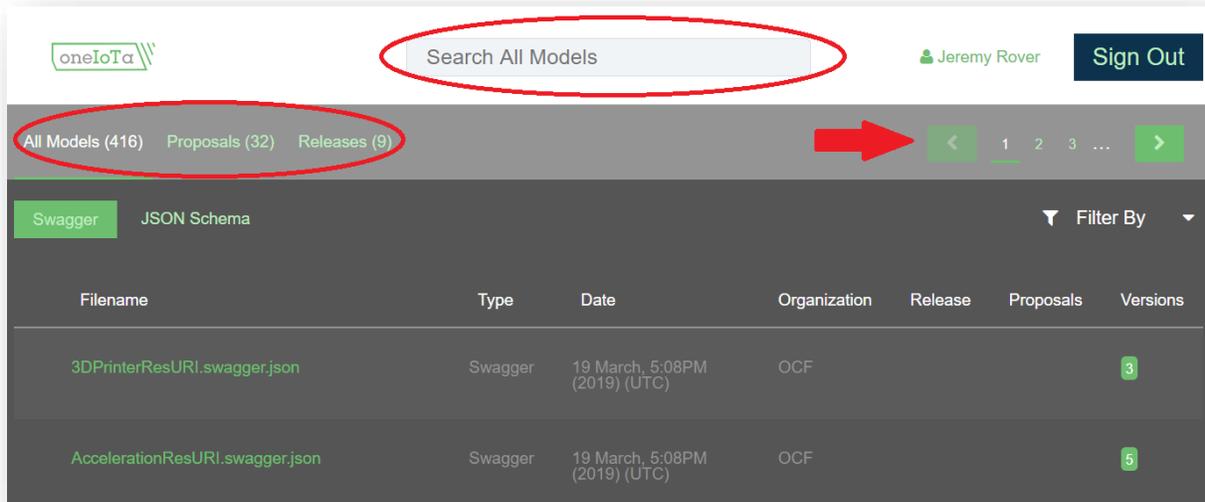
Callouts:

- All Models (117):** Database of files; Comprehensive search feature; Toggles between RAML, JSON, or All
- Proposals (12):** Proposal: Must accept license agreement and have 'Submitter' status*; Allows you to enter multiple JSON and RAML files under a single proposal; Save as a Draft before submitting it for approval.
- Releases (0):** Releases: Shows approved proposals, by organization.
- Sign Out:** Update your user account information

All Models

The *All Models* area is a list of the individual data model files and can be *Filtered By* approval status or organization. The header search feature allows users to search based on specific data model types (e.g. light, thermostat, etc.).

Users with *Viewer* status are able to sign on to the site and see this information, and copies of the approved OAS 2.0 released files are also saved on the OCF GitHub repository: <https://github.com/openconnectivityfoundation/loTDataModels>. Users will be able to see the specific revision numbers for each data-model in the far right column titled *Version*.



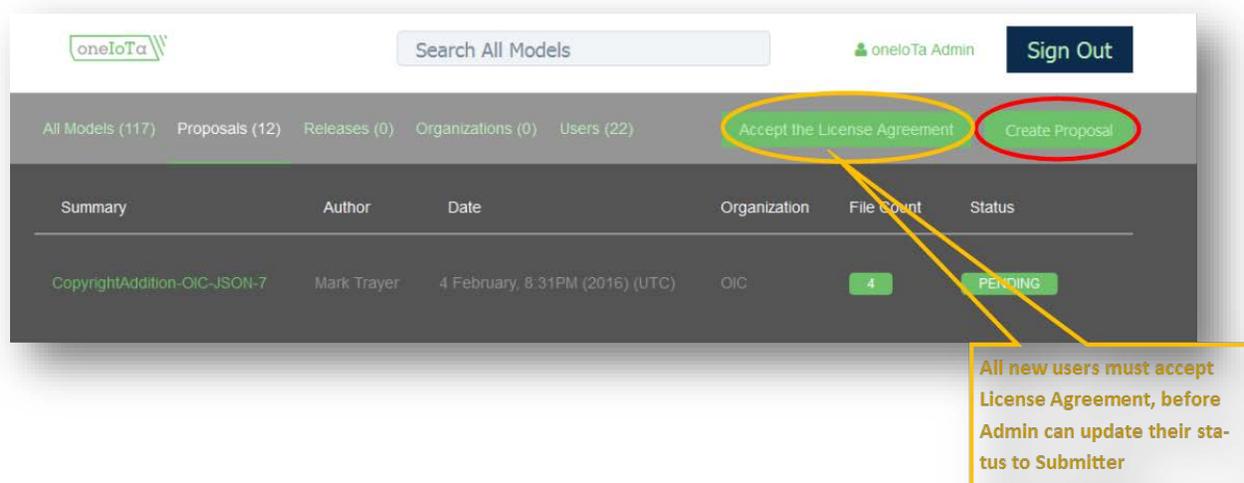
The screenshot shows the oneloTa website interface. At the top, there is a search bar labeled "Search All Models" and a user profile for "Jeremy Rover" with a "Sign Out" button. Below the search bar, there are navigation links for "All Models (416)", "Proposals (32)", and "Releases (9)". A red arrow points to a pagination control showing "1 2 3 ...". Below this is a table with columns: Filename, Type, Date, Organization, Release, Proposals, and Versions. The table contains two rows of data for Swagger files.

Filename	Type	Date	Organization	Release	Proposals	Versions
3DPrinterResURI.swagger.json	Swagger	19 March, 5:08PM (2019) (UTC)	OCF			3
AccelerationResURI.swagger.json	Swagger	19 March, 5:08PM (2019) (UTC)	OCF			5

Proposals

Users with *Submitter* status will be able to login to the oneloTa site, click on the *Proposals* area, and upload data models directly from their desktop to the site. If a user only has *Viewer* status, they must click on the *Proposals* area, and accept the **Contributor Agreement**. Admin will then update the user's status to *Submitter* (this could take 24-48 hours).

Once a user receives *Submitter* status, that user can create proposals. To do so, the user should click on the **Create Proposal** button in the upper right hand side of the screen (see below).



The screenshot shows the oneloTa website interface for the 'Proposals' section. At the top, there is a search bar labeled "Search All Models" and a user profile for "oneloTa Admin" with a "Sign Out" button. Below the search bar, there are navigation links for "All Models (117)", "Proposals (12)", "Releases (0)", "Organizations (0)", and "Users (22)". Two buttons are highlighted: "Accept the License Agreement" (circled in yellow) and "Create Proposal" (circled in red). Below this is a table with columns: Summary, Author, Date, Organization, File Count, and Status. The table contains one row of data for a proposal.

Summary	Author	Date	Organization	File Count	Status
CopyrightAddition-OIC-JSON-7	Mark Trayer	4 February, 8:31PM (2016) (UTC)	OIC	4	PENDING

All new users must accept License Agreement, before Admin can update their status to Submitter

If you have any questions, please contact oneiota@openconnectivity.org

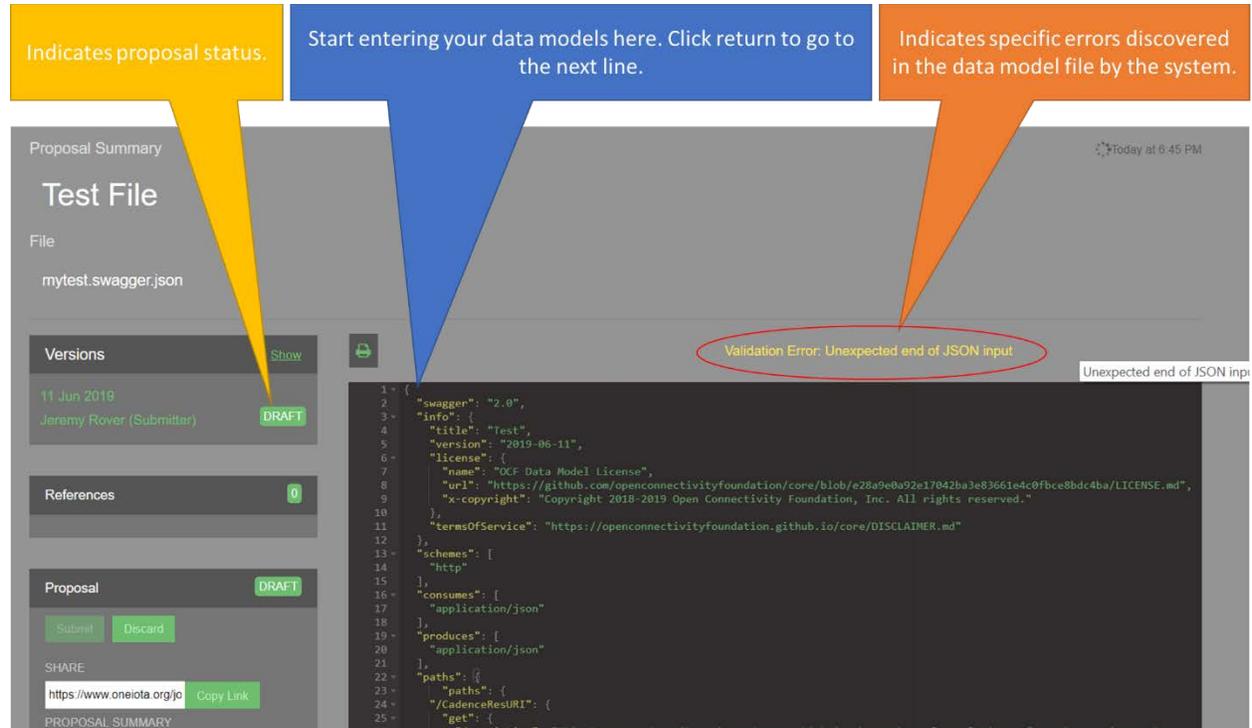
Entering Data Models:

A *Submitter* just clicks on the “Create Proposal” button to create a new model. A proposal can contain one or more models. The “Proposal Summary” is an optional description field that will eventually show up in the git repository if the proposal is approved. When a file name is added (all files must either have the file extension “.json”), the file is automatically created and an editing window is opened. The submitter can simply begin entering code, start with a default template, or paste in copied text. Files can also be batch-imported via an import window that allows for the selection of multiple files in a single import instance (see upload data models section below). If a file being added to or imported to a proposal already exists then oneIoTa automatically creates a new version of that file.

Tip: Users can, and are encouraged to, review other data models on the oneIoTa site, to understand the recommended syntax to obtain proposal approval. If you have specific questions about approved data model syntax, please email oneiota@openconnectivity.org.

Note: File names must end in .swagger.json for the site to allow the proposal to be created.

Once the file has been created, you will see the proposal info at the top of the page, the file name just below that, and you will be able to start entering code into the black window to the right of the proposal information.



The screenshot shows the oneIoTa interface. On the left, there is a sidebar with sections: 'Versions' (showing a draft by Jeremy Rover), 'References' (0), and 'Proposal' (DRAFT). The main area is titled 'Test File' and shows the file 'mytest.swagger.json'. A code editor on the right contains Swagger JSON code. An orange callout bubble points to a red circle around the text 'Validation Error: Unexpected end of JSON input' in the code editor. Another orange callout bubble points to a 'DRAFT' status indicator. A blue callout bubble points to the code editor area with the text 'Start entering your data models here. Click return to go to the next line.' A yellow callout bubble points to the 'DRAFT' status indicator with the text 'Indicates proposal status.'

Once you start entering information into the data model entry window, users will see orange text appear just above the window. This alerts the submitter of possible errors in the data

If you have any questions, please contact oneiota@openconnectivity.org

model code that the tool is detecting. The edit window understands the correct syntax of the RAML and JSON schema files, and will color-code and validate the syntax as the User enters it.

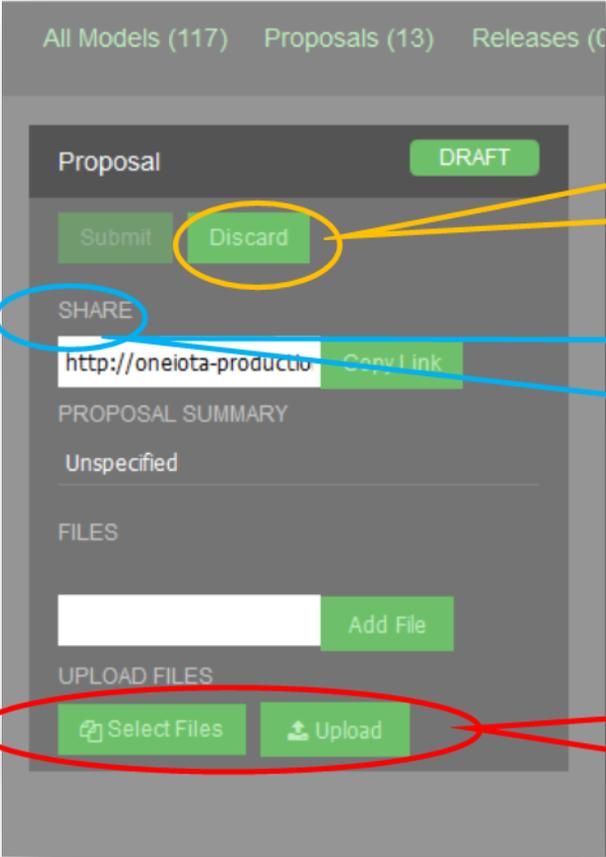
Note: Data models will save automatically and the date/time-stamp in the upper right corner of the screen indicates the last time the file was saved.

Uploading Data Models:

To upload files directly from your desktop, open a new proposal, click **Select Files**, which will open a dialog box on your screen, and search for the specific files. Users can filter files by 'all', '.json', or '.swagger.json' (users should only upload '.swagger.json' files) then select the files required and 'open.' Next you will need to click **Upload**, at which point all of the files will be uploaded to the proposal.

If you are working from a file you originally downloaded from the oneloTa site¹, select the .swagger.json file with a name matching the existing .swagger.json file in the data model, click Upload, and the tool will create a new version of the existing proposal using the uploaded file.

¹ See "Download Data Models" section below



The screenshot shows a mobile application interface for managing proposals. At the top, there are navigation links for 'All Models (117)', 'Proposals (13)', and 'Releases (0)'. The main content area is titled 'Proposal' and includes a 'DRAFT' status indicator. Below this, there are 'Submit' and 'Discard' buttons. A yellow callout box points to the 'Discard' button with the text: 'Discard draft. A window will pop up asking you to confirm your request. Click "ok" to accept, or cancel to "cancel."' Below the buttons is a 'SHARE' section with a text input field containing a URL and a 'Copy Link' button. A blue callout box points to the 'Copy Link' button with the text: 'Share the open proposal with fellow oneIoTa users by clicking "Copy Link" and sending it to them. Users with Viewer status are not able to submit proposal, but they can assist with creating them'. Below the share section is a 'PROPOSAL SUMMARY' section with the text 'Unspecified'. Underneath is a 'FILES' section with an 'Add File' button. At the bottom, there is an 'UPLOAD FILES' section with 'Select Files' and 'Upload' buttons. A red callout box points to these two buttons with the text: 'Upload files directly from your own desk top by clicking "Select Files," select the files you wish to upload, then click Upload.'

Sharing Proposals:

Submitters can also “Share” their proposals with other users on the oneIoTa site. To do this, the *Submitter*, will need to copy the share link from the left hand navigation bar, and send it to any other users they wish to share it with. Submitters are able to share draft proposals with users, even if they only have *Viewer* status, but only *Submitters* are able to submit the proposal for *Reviewer* approval.

TIP: Submitters who share draft proposal with other users, may want to be cautious about who is editing files, as it is their responsibility as the Submitter, to make sure the proposals is compliant. It is recommended to work offline, if consulting with multiple users, then uploading all files at once to the proposal area for final review before submitting.

Submitting Proposals:

After the user is satisfied with their draft proposal, they will need to click **Submit**. Depending on the organization to which the *Submitter* is submitting the proposal, the *Reviewer* of that organization will receive an email notification that a new proposal has been submitted, and

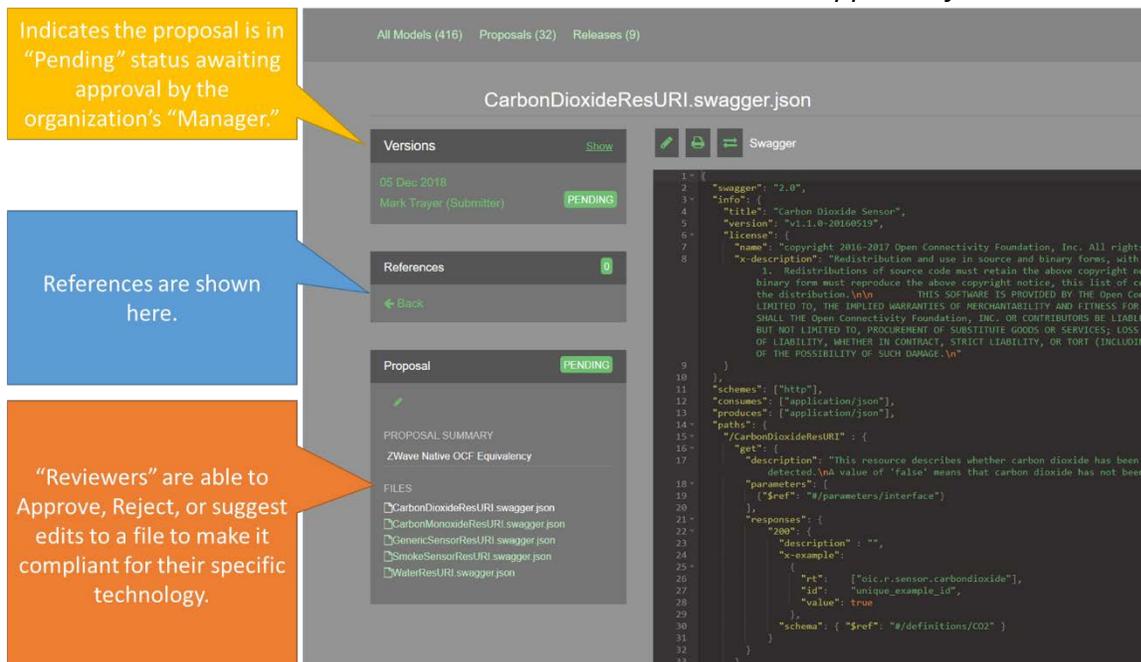
requires their review, prior to approval. Proposals can be submitted to multiple organizations at once, but all proposals must be compliant with OAS 2.0 schema as determined by OCF.

Note: OCF is the only accepting organization at the moment, so all submissions must comply with OCF specifications.

Once the proposal is submitted, the *Reviewer* of the organization will review the proposal. Within the *Reviewer's* window, you are able to see which organization the submission was submitted to, and the list of files associated with each proposal in the left-hand side bar. *Reviewers* can suggest edits within each data model file of the proposal. When a proposal is in the pending status, the *Submitter* is unable to edit it File references can be followed by clicking them in the reference box.

Reviewers can navigate through each file separately, and return to the previous file by hitting the **← Back** button in the left-hand side bar. Once a *Reviewer* has finished reviewing all files within the proposal, they can either select to **Approve** or **Reject** the proposal.

Please Note: *If a proposal contains several files, all files must be accepted or rejected at the SAME TIME. Each organization will be responsible for creating their own internal checks and balances to make sure data models with incorrect code are not approved for release.*



Indicates the proposal is in "Pending" status awaiting approval by the organization's "Manager."

References are shown here.

"Reviewers" are able to Approve, Reject, or suggest edits to a file to make it compliant for their specific technology.

```

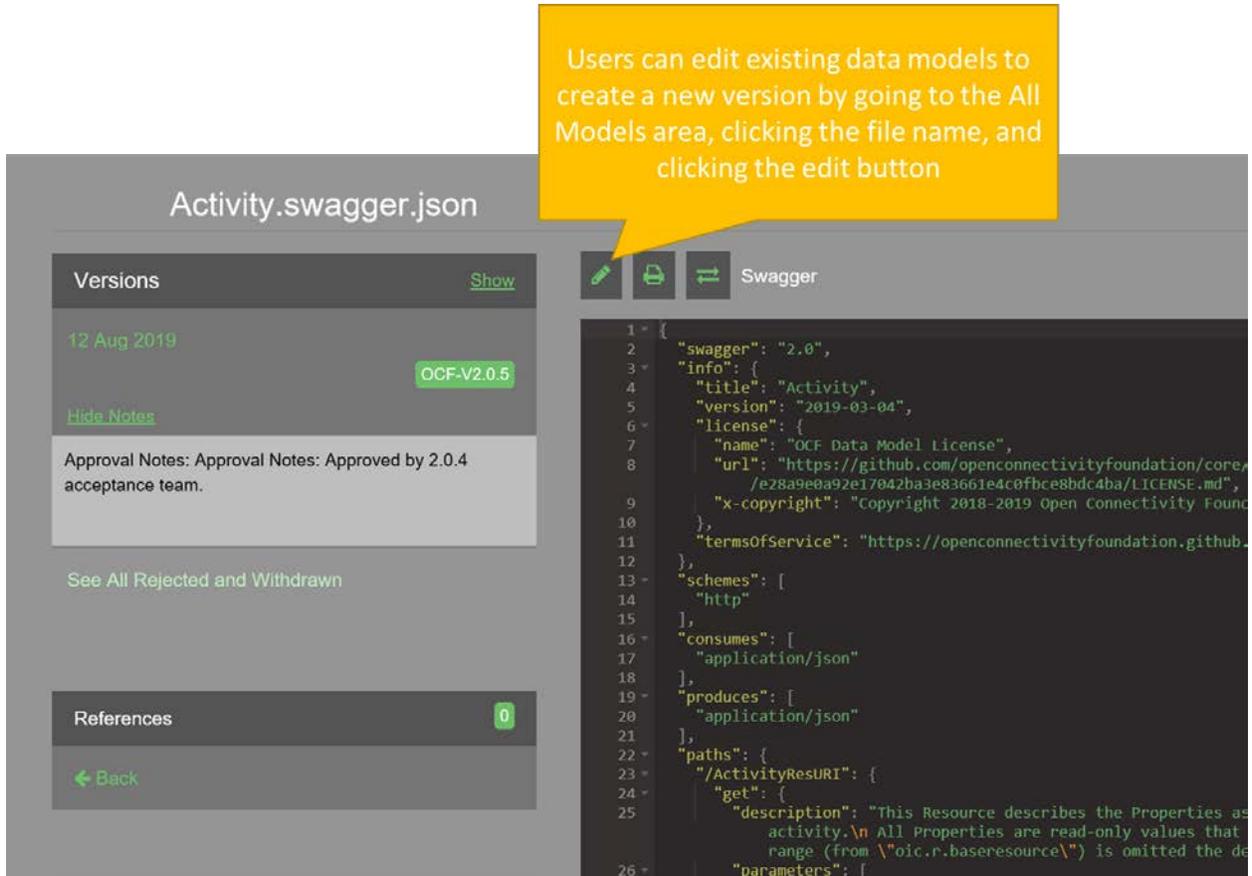
1 {
2   "swagger": "2.0",
3   "info": {
4     "title": "Carbon Dioxide Sensor",
5     "version": "v1.1.0-20100519",
6     "license": {
7       "name": "Copyright 2016-2017 Open Connectivity Foundation, Inc. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the source code and any accompanying documentation are distributed under the same license as the original work. THIS SOFTWARE IS PROVIDED BY THE OPEN CONNECTIVITY FOUNDATION, INC. OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY KIND, INCLUDING BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION, HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE."
8     }
9   },
10  "schemes": ["http"],
11  "consumes": ["application/json"],
12  "produces": ["application/json"],
13  "paths": {
14    "/CarbonDioxideResURI": {
15      "get": {
16        "description": "This resource describes whether carbon dioxide has been detected. A value of 'false' means that carbon dioxide has not been detected.",
17        "parameters": [
18          {
19            "name": "interface",
20            "in": "query",
21            "required": true,
22            "description": "The interface to use for the sensor.",
23            "x-example": "0x10101010"
24          }
25        ],
26        "responses": {
27          "200": {
28            "description": "",
29            "x-example": {
30              "value": true
31            }
32          }
33        }
34      }
35    }
36  }
37 }

```

Once a *Reviewer* approves the submission, the proposal will be placed in a release queue. The *Reviewer* will then need to go to the Release area and tag the approved submission. If the

If you have any questions, please contact oneiota@openconnectivity.org

Reviewer rejects the data model, a notification email will be sent to the submitter notifying them of the reviewer's decision, and when applicable, revision suggestions.



Activity.swagger.json

Versions [Show](#)

12 Aug 2019 OCF-V2.0.5

[Hide Notes](#)

Approval Notes: Approval Notes: Approved by 2.0.4 acceptance team.

[See All Rejected and Withdrawn](#)

References 0

[← Back](#)

Swagger

```
1 {
2   "swagger": "2.0",
3   "info": {
4     "title": "Activity",
5     "version": "2019-03-04",
6     "license": {
7       "name": "OCF Data Model License",
8       "url": "https://github.com/openconnectivityfoundation/core/
9         /e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LICENSE.md",
10      "x-copyright": "Copyright 2018-2019 Open Connectivity Found
11    },
12    "termsOfService": "https://openconnectivityfoundation.github.
13  },
14  "schemes": [
15    "http"
16  ],
17  "consumes": [
18    "application/json"
19  ],
20  "produces": [
21    "application/json"
22  ],
23  "paths": {
24    "/ActivityResURI": {
25      "get": {
26        "description": "This Resource describes the Properties as
27          activity.\n All Properties are read-only values that
28          range (from \"oic.r.baseresource\") is omitted the de
29        "parameters": [
```

Approval Process

OCF's IoT Data Modeling *Reviewer* group will periodically review the pending models and determine if the proposal should be approved. If approved, no status indicator will show and the proposal will be pushed to the git repository. At that point, it can be pulled from the git repository or viewed in oneIoTa, but it can no longer be edited. It will be tagged as part of the next release unless a new proposal is created and accepted before the release is tagged.

Releases

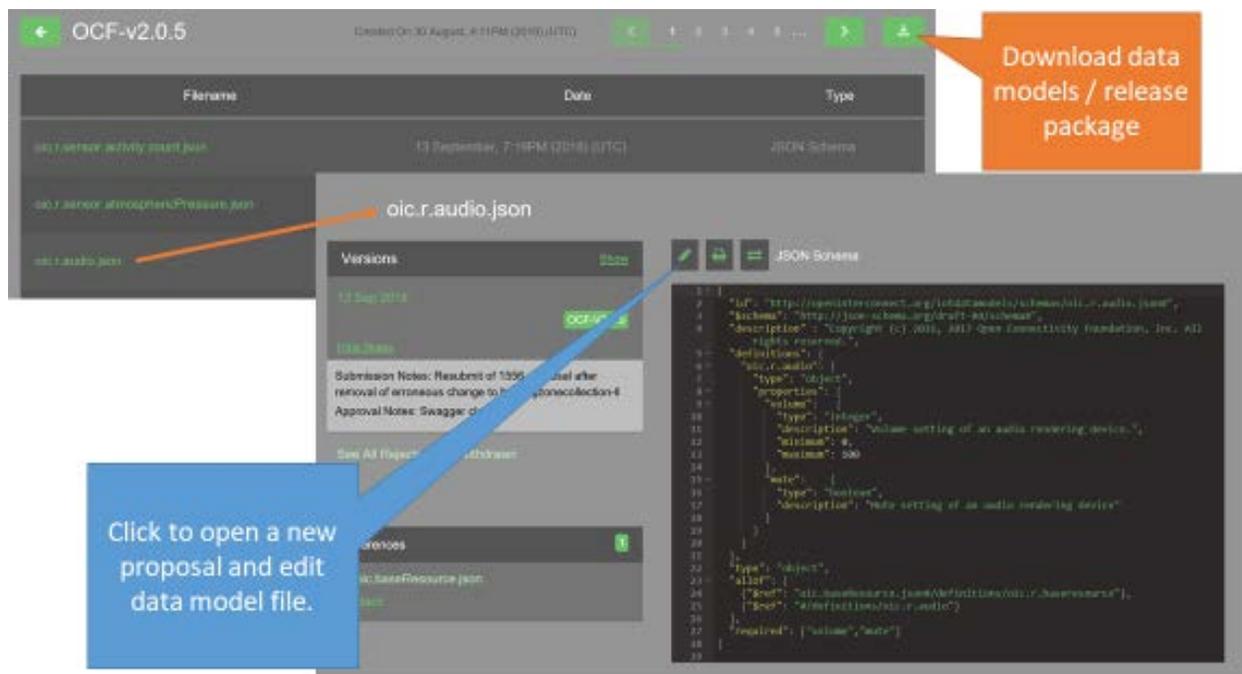
To release an approved proposal *Reviewers* must tag the proposal, using a specific naming convention as directed below, and only under their own organization's name. Once the proposal is released, it will appear in the "Release Area." Once a proposal has been approved, it will be pushed to the OCF GitHub repository, where users will be able to download it for their own use.

If you have any questions, please contact oneiota@openconnectivity.org

Note: The “Proposal Summary” information is no longer showing as a reference field.

Downloading and Editing Released Data Models:

User are able to edit existing data model files by downloading them directly to their own desktop or within the oneloTa tool. To download a released data model, users must click on the released data model, then on the download button in the right hand corner. Users will then be prompted to accept the **License Agreement**. The user will need to click the download button again, after accepting the **License Agreement**, which will open a window to download the selected data model files in a zip file format.



The screenshot displays the oneloTa interface for OCF-v2.0.5. At the top, a table lists data models with columns for Filename, Date, and Type. An orange callout box points to a download icon in the top right corner, labeled "Download data models / release package". Below the table, a specific model "oic.r.audio.json" is selected, and its details are shown in a modal window. A blue callout box points to the "Versions" section of this modal, labeled "Click to open a new proposal and edit data model file.". The modal also shows a "JSON Schema" editor with a code editor displaying the schema definition for "oic.r.audio".

If a user is wants to edit an existing data model file they will click on the release, then on the specific file name. Once the data model file is opened, the user can click the edit button at the top left corner of the data model code window which will open up a new draft proposal with the same data model code and file name as the prior data model file.

Git Repository

The oneloTa tool only commits to and tags a git repository master. oneloTa maintains a separate internal database for its functionality. The approving organization will determine

If you have any questions, please contact oneiota@openconnectivity.org



which repository it wants to use. In the case of OCF, the repository is open to all for pulling the models, but contributions back to the master must be done through oneIoTa.

If you have any questions, please contact oneiota@openconnectivity.org

Derived Models

One of the major obstacles with the Internet of Things is that there are many incompatible ecosystems. While some have addressed this problem by writing various types of converters between ecosystems, this becomes hard to scale. The OCF architecture and the oneIoTa tool use OCF data models as a “common” data model and a derived version of these models to define conversions between OCF and other IoT ecosystems. This makes all derived models interoperable with OCF and all other derived models through (at most) two conversion steps. As with all OCF data models derived models can be machine-read to automatically create code stubs.

Derived models use standard JSON schema syntax. Fundamentally, derived models provide a conversion mapping between OCF data models and similar data models in another IoT ecosystem. These conversions can be very simple (as in just a property name conversion) to extremely complex (as in converting between different numbers of properties with complex mathematics). Examples of the various conversions are described below.

Simple Mapping

Simple mapping just converts between different field names. Simple mapping accounts for field mappings between ecosystems. It can be used in combination with direct mappings as well as more complex mathematical conversions. In the example below, the derived model defines the field “lightness” to map to the OCF field “brightness.” It also maps the derived ecosystem field “rgb_color” to the three OCF fields “red,” “green,” and “blue.”

```
{
  "properties": {
    "lightness": {
      "type": "number",
      "oic_conversion": {
        "oic-alias": "brightness"
      }
    },
    "rgb_color": {
      "type": "string",
      "oic_conversion": {
        "oic_alias": ["red", "blue", "green"]
      }
    }
  }
}
```

Conversion

Conversion defines mathematical conversion between simple mappings. It does this using the two fields “x-from-ocf” and “x-to-ocf.” The mathematical conversion is defined in a string. The string is not validated. In this example, the field “darkness” in the derived model is converted to the field “brightness” in the OCF model by subtracting it from 255. A script that creates code stubs could read this model to identify input and output variables and use the conversion strings as comments that a coder could reference to properly implement the conversions in any programming language.

The field “darkness” is defined in the other ecosystem and mapped. “Brightness” should be prefixed with the resource type.

```
{
  "properties": {
    "darkness": {
      "type": "number",
      "x-ocf-conversion": {
        "x-to-ocf": "oic.r.light.brightness.json = 255 - darkness",
        "x-from-ocf": "darkness = 255 - oic.r.light.brightness.json"
      }
    }
  }
}
```

The following example is a bit more complex and demonstrates the use of a list of strings. The derived model field “darkness_percentage” is converted to and from the OCF field brightness with two conversion steps for each direction.

```
{
  "properties": {
    "darkness_percentage": {
      "type": "number",
      "ocf_conversion": {
        "to_ocf": [
          "darkness = 255 * darkness_percentage",
          "brightness = 255 - darkness"
        ],
        "from_ocf": [
          "darkness = 255 - brightness",
          "darkness_percentage = darkness_percentage / 255"
        ]
      }
    }
  }
}
```

```
}
```

Additional Derived Model Examples

For additional examples of derived models, see the following github repositories:

<https://github.com/openconnectivityfoundation/OCF-Zigbee>

<https://github.com/openconnectivityfoundation/OCF-oneM2M>

<https://github.com/openconnectivityfoundation/UPnP-models>

Derivative Model Validation

As with common “OCF” models, derived models are validated against JSON syntax. In addition, derived models are validated to ensure they are properly referenced to an OCF model. The derived model must include all the fields of the OCF model from which it is derived. These fields consist of x-ocf-alias, x-to-ocf, and x-from-ocf within the x-ocf-conversions property.

```
"properties": {
  "printType": {
    "type": "string",
    "description": "3D Printer Type",
    "x-ocf-conversion": {
      "x-ocf-alias": "oic.r.3dprinter",
      "x-to-ocf": [
        "oic.r.3dprinter.3dprinttype = printType"
      ],
      "x-from-ocf": [
        "printType = oic.r.3dprinter.3dprinttype"
      ]
    }
  }
}
```

This means that if a relevant model does not yet exist in OCF, it must be proposed and accepted by OCF before a derived model can be created in some other ecosystem. This expands the models within OCF and guarantees that the derived model will work with OCF models and all other derived models.

What's Next?

Derived models in OCF are an extremely powerful solution to interoperability between ecosystems in the Internet of Things. They are also very flexible in being able to support RESTful as well as other architectures.

Other ecosystems interested in using oneIoTa and adding their models to the interoperable OCF ecosystem should contact OCF.