OEM Dashboard

Configuration Guide

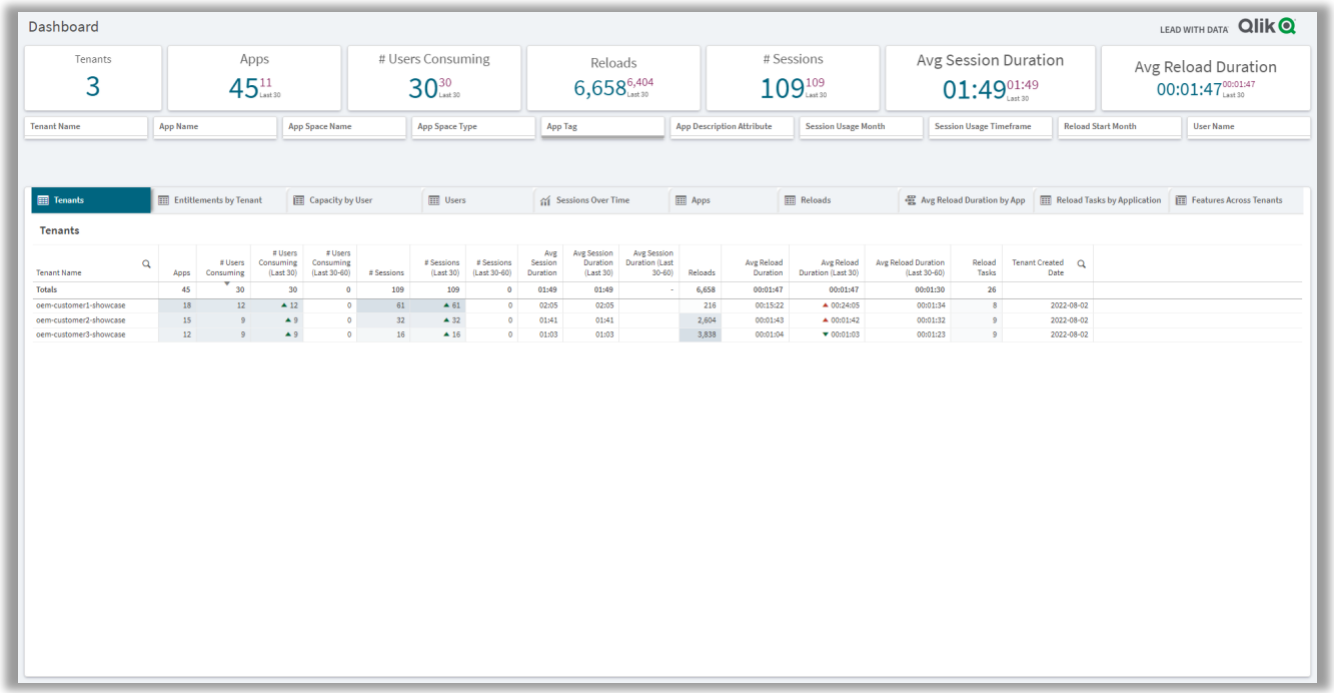
v2.0.0

TABLE OF CONTENTS

Introduction

The *OEM Dashboard* is an application for Qlik Cloud designed for OEM partners to centrally monitor data across their customers' tenants. It provides a single pane to review numerous dimensions and measures, compare trends, and quickly spot issues across many different areas—which would otherwise be a tedious process. This application includes data from the *App Analyzer*, *Entitlement Analyzer*, and the *Reload Analyzer*, all of which are other monitoring applications for Qlik Cloud that provide deep levels of detail on their respective areas. Together, a complete picture can be formed which is crucial to the successful management of an OEM environment.

This guide provides an overview of the requirements and process of configuring the *OEM Dashboard* application for Qlik Cloud. This guide does not cover the automated deployment of this application, but rather shows how the process is done manually.



Pre-Requisites

Required Monitoring Applications

The following monitoring applications are required on all tenants. This guide will reference how to manually set them up, however in a production environment at scale, they can be automatically configured using Qlik's APIs. These same applications are used across all tenants in this guide and are configured differently based on the purpose of the tenant that they are installed on.

App Analyzer – provides application metadata.

Entitlement Analyzer – provides entitlement, user, and usage data.

Reload Analyzer – provides reload, tasks, and data connection data.

Each of the applications requires (once per tenant, can be shared across applications):

- A user with **TenantAdmin** and **Developer** roles
- An API Key (for the aforementioned user)
- A REST Connector configured to interact with the tenant's APIs

Third-Party Storage

As of the time of this guide, third-party storage is required for the centralization of data so that it can be loaded into a single tenant. All tenants will write and read from the same location, allowing the sharing of data across.

Tested Storage Connections

- Amazon S3
- Google Cloud Storage
- Azure Storage

Terminology & Overall Flow

Terminology

The terms “Parent” and “Child” are used throughout this guide to denote the OEM Partner (“Parent”) and Customer (“Child”) tenants. There are there no technical differences in the capabilities of these tenants (tenants are non-hierarchical). This terminology, however, helps to separate where the OEM Partner’s control and internal analysis lives as compared to their customers’ users. It is assumed that each OEM partner will have at least one of these “Parent” tenants, as they will likely not be repurposing one of their customers’ tenants for management and internal analysis.

As the terminology pertains to this guide:

Parent Tenant: This is the tenant that will host the monitoring applications in “Parent” mode as well as the *OEM Dashboard*. These applications load from the data generated by all “Child” tenants’ monitoring applications.

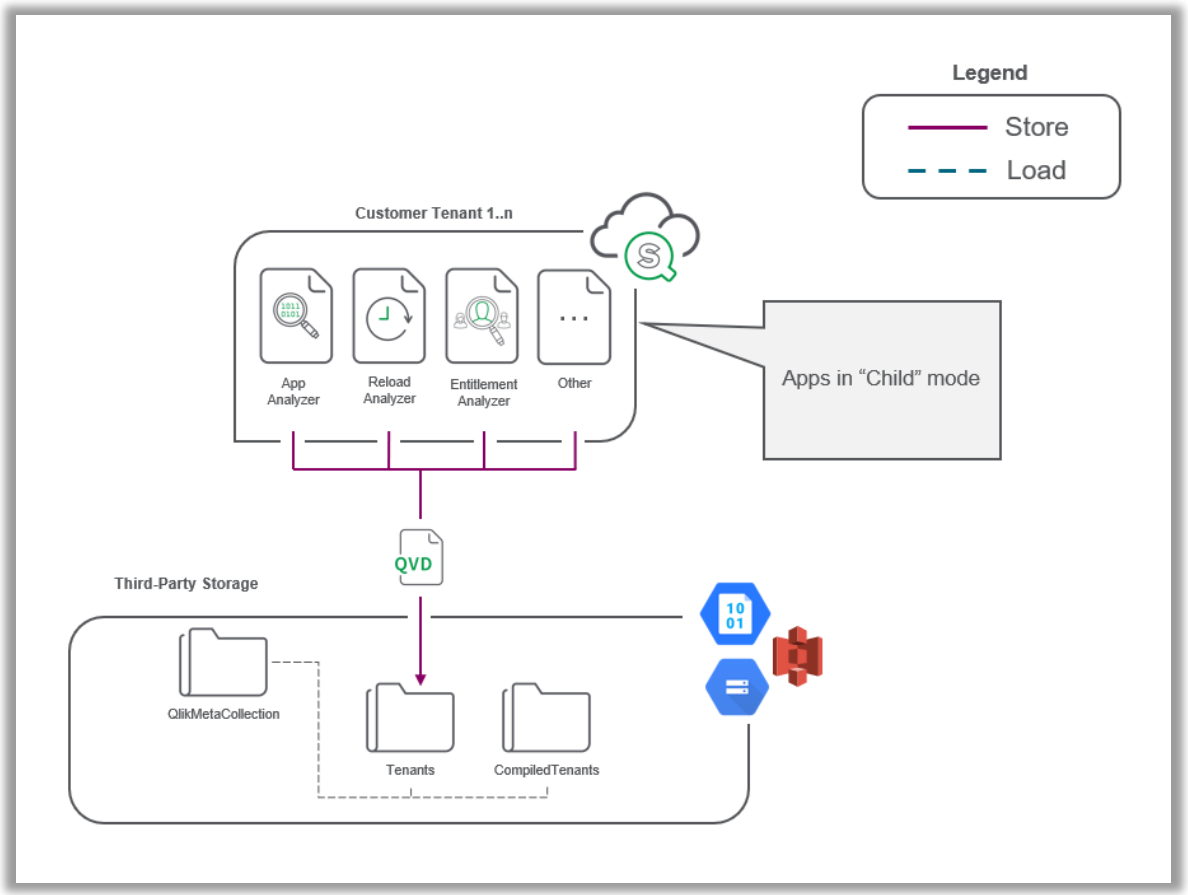
Child Tenant: These are the customer tenants that each have the monitoring applications installed on them in “Child” mode.

Monitoring Apps: This term refers to the core monitoring applications, including the *App Analyzer*, *Reload Analyzer*, and *Entitlement Analyzer*. All three of these applications can run in “Parent”, “Child”, or “normal” mode (“normal” being non-OEM with the capability flags toggled off).

Third-Party Storage: The centralized storage where all the metadata across all tenants will be located.

Overall Flow

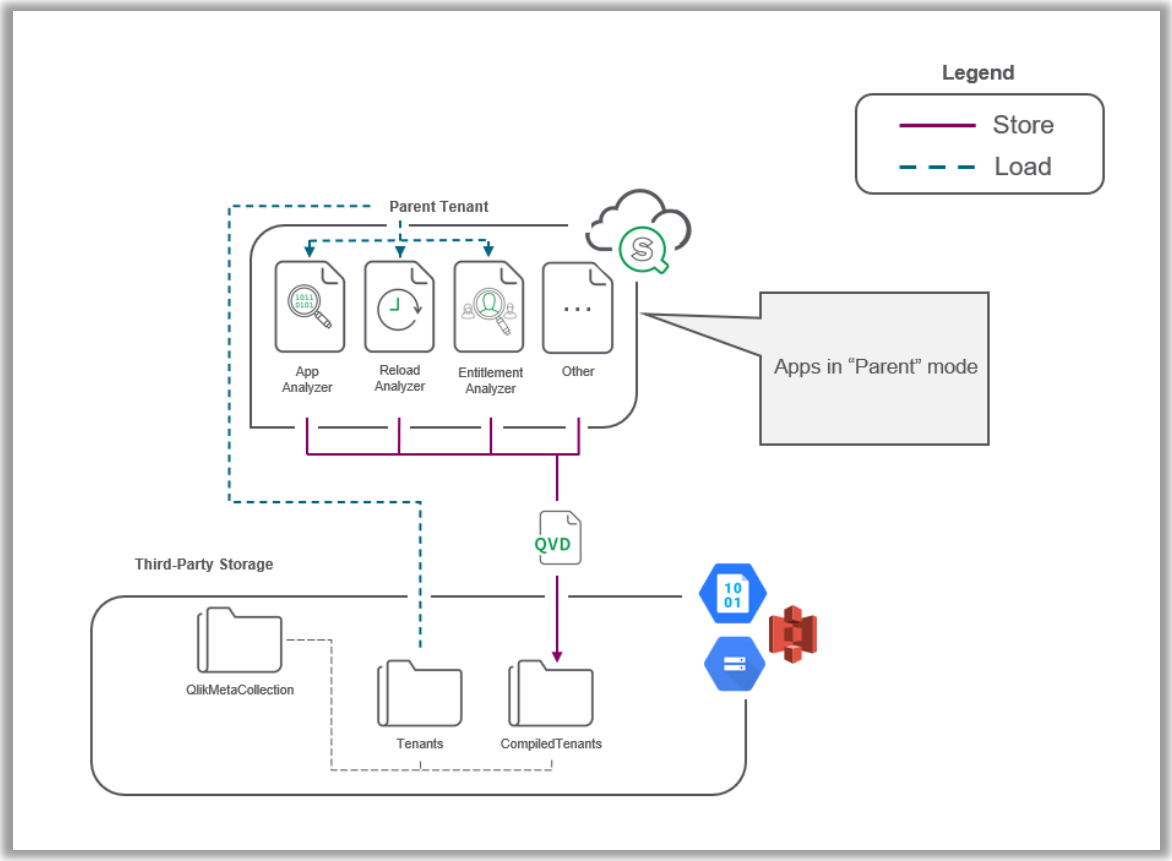
Step 1



All monitoring applications are installed and configured on the “Child” tenants are set to “Child” mode in the load script.

When they reload, their QVDs (non-transformed) are stored to centralized storage within the `Tenants` directory for each tenant ID, respectively. The folder structure is created automatically.

Step 2

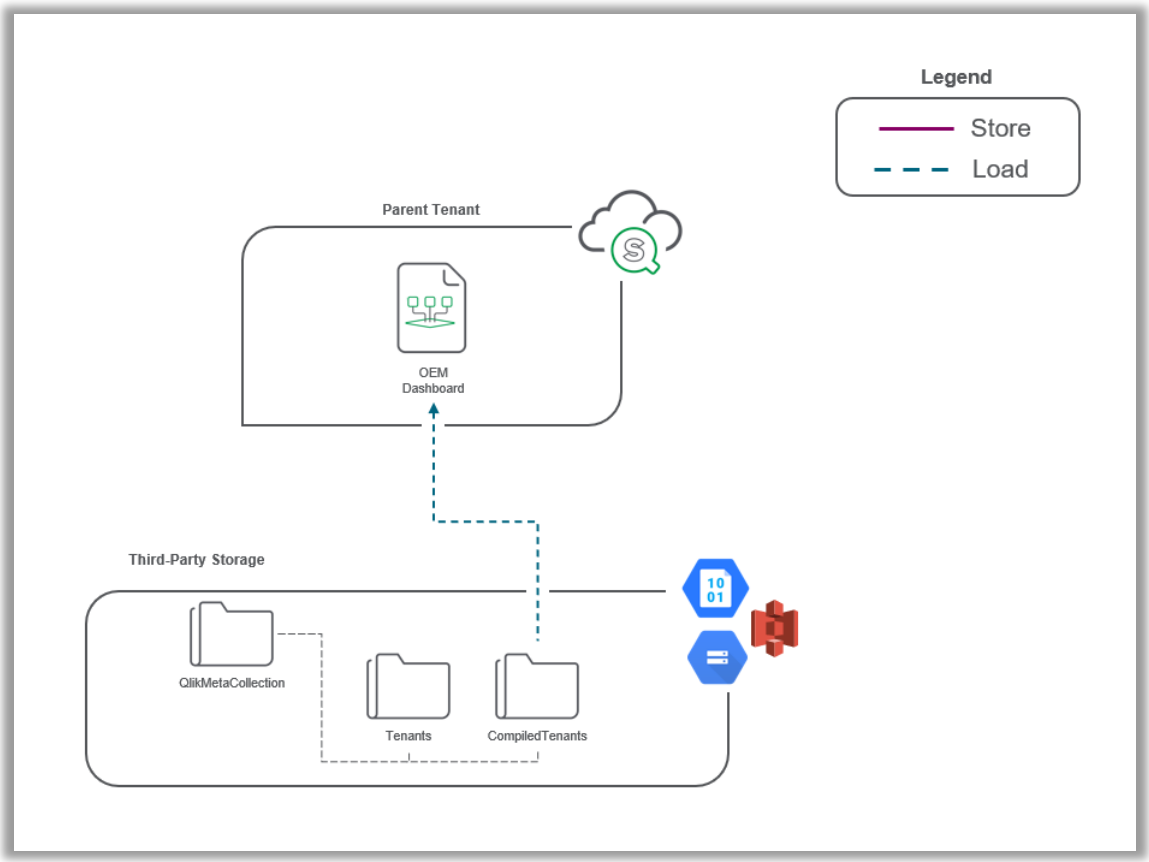


The monitoring applications are installed and configured on the “Parent” tenant are set to “Parent” mode in the load script.

When they reload, they load from all “Child” directories, concatenate the models, and then perform any transformations necessary.

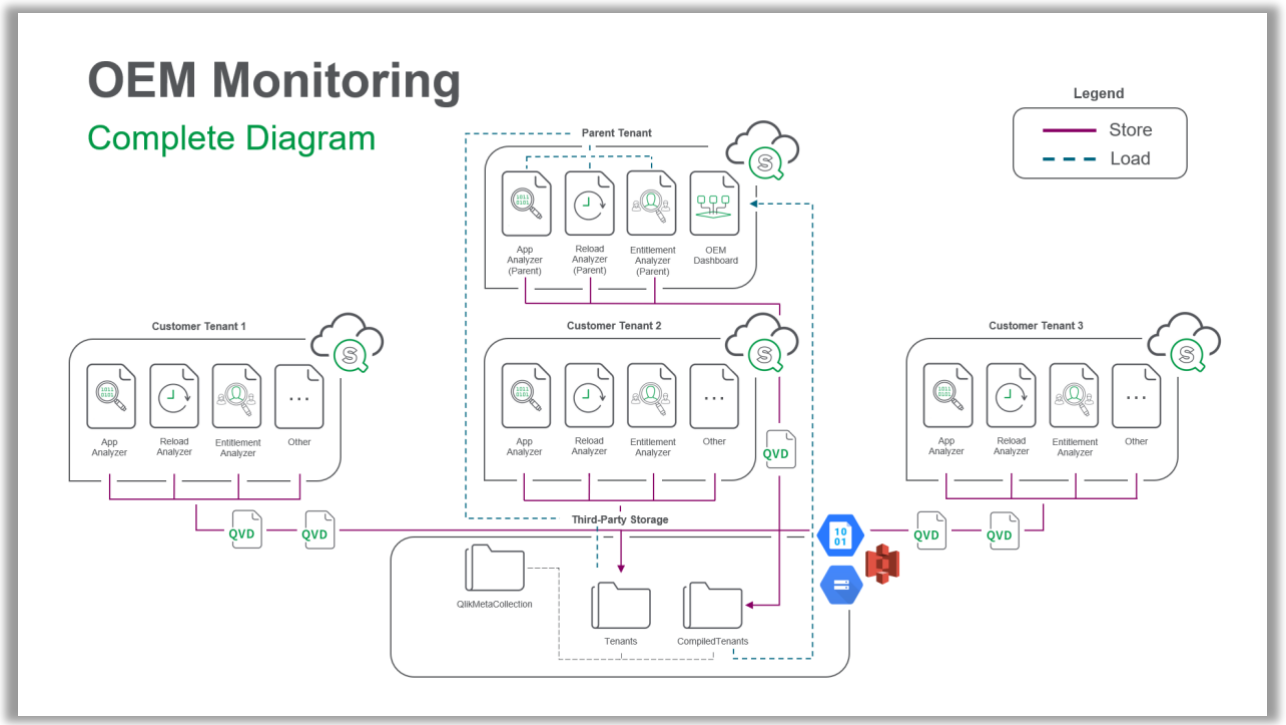
At the end of the reload of each “Parent” application, they store the final, composite model to QVD in the `CompiledTenants` directory. The folder structure is created automatically.

Step 3



The *OEM Dashboard* reads in the required data from the `CompiledTenants` directory, combining the necessary tables from all models into a single model.

Overview


OEM Monitoring
Complete Diagram
Legend
Store
Load
Parent Tenant
App Analyzer (Parent)
Reload Analyzer (Parent)
Entitlement Analyzer (Parent)
OEM Dashboard
Customer Tenant 1
Customer Tenant 2
Customer Tenant 3
App Analyzer
Reload Analyzer
Entitlement Analyzer
Other
QVD
Third-Party Storage
QlikMetaCollection
Tenants
CompiledTenants

Data Structure: QlikMetaCollection

All metadata which is stored and read from the monitoring applications and *OEM Dashboard* is housed under the `QlikMetaCollection` directory within the designated third-party storage provider. This directory can reside beneath any number of parent directories, and the structure beneath `QlikMetaCollection` it is automatically generated (including the `QlikMetaCollection` directory itself). It is important to state that the underlying directories should **not** be tampered with, as the monitoring applications are dependent on its structure.

After the completion of the monitoring application reloads, the structure will resemble the following (the `Settings` directories will appear if the *Console Settings Collector add-on* is being leveraged):

```
QlikMetaCollection/
    ↳ Tenants/
        ↳ <TenantID>/
            ↳ Monitoring/
                ↳ <MonitoringApp>/
                    ↳ <QVDs>
            ↳ Settings/
                ↳ <QVDs>/
    ↳ CompiledTenants/
        ↳ Monitoring/
            ↳ <MonitoringApp>/
                ↳ <QVDs>
        ↳ Settings/
            ↳ <QVDs>
```

Notice that the `CompiledTenants` directory is one level shallower than the `Tenants` directory. This is because all “Child” tenant data is consolidated by the “Parent” monitoring applications. This directory is what the *OEM Dashboard* ingests.

Configuration

Summary Table

Once the configuration is completed, the final deployment should resemble the below:

	Parent Tenant	Child Tenants
OEM Dashboard	✓	
App Analyzer	✓	✓
Entitlement Analyzer	✓	✓
Reload Analyzer	✓	✓
Console Settings Collector *optional	*✓	*✓

"Child" Tenant Monitoring Application Configuration (for each "Child" Tenant)

1. Create a connection to a third-party storage provider found in the Third-Party Storage section above. This connection can reside in the same **Shared** space as the monitoring applications, or a separate space, just be aware that the default placeholders for the connection names and QVD storage locations use relative paths (more below).

2. Download the below applications and the accompanying *Qlik Cloud Monitoring Applications Installation Guide* (these same applications are used for the “Parent” tenant as well):
    - *App Analyzer*
    - *Entitlement Analyzer*
    - *Reload Analyzer*

3. Import each application into the desired **Shared** space.
4. Follow the **Configure the Tenant for Monitoring Applications** section from the installation guide to setup the below requirements (this only needs to be done once):
    a. A user with **TenantAdmin** and **Developer** roles
    b. An **API Key** (for the aforementioned user)
    c. A **REST Connector** configured to interact with the tenant’s APIs named **monitoring_apps_REST**
5. Navigate to the load script of each application and make the following adjustments:
    a. Navigate to the *** Optional Configuration *** tab. Find the variable `vu_multi_tenant_enabled` and set it to `1`. This toggles on multitenant mode. Leave the `vu_is_parent_app` variable set to `0`, as that will only be set on “Parent” tenants. You will modify this step as per the instructions in the “Parent” Tenant Monitoring App Configuration section below.

	Parent Tenant	Child Tenants
`vu_multi_tenant_enabled`	1	1
`vu_is_parent_app`	1	0

b. Remaining on the same tab, find the `vu_qlik_meta_collection_parent_dir` variable and enter the location where the `QlikMetaCollection` folder resides within your third-party storage. This folder is responsible for housing all metadata across all tenants. If this folder does not exist, it will be automatically created in the location that you enter. For example, if the desired directory structure should resemble `<Connection>/Folder1/Folder2/QlikMetaCollection`, then you would enter:

```
SET vu_qlik_meta_collection_parent_dir = 'lib://:<Connection>/Folder1/Folder2;
```

If the directory is not to be beneath any other folders, then you would enter either:

```
SET vu_qlik_meta_collection_parent_dir = 'lib://:<Connection>;
```

A trailing slash and space-relative semi-colon are both optional.

c. **For the *Reload Analyzer* Only:** It is important to consider how far back the data is fetched and for how long it is held (rolling timeframe). Variables that control these periods can be found within this same script tab and are `vu_initial_days_back` and `vu_reload_rolling_range`, respectively. Depending on how many "Child" tenants there are, the number of applications, the amount of expected reloads, etc. – you will want to make sure that these ranges aren't too wide (e.g., not 365). It is important to consider that the *OEM Dashboard* views most metrics in rolling periods (last 30 days rolling and last 30-60 days rolling), so perhaps an appropriate setting for each variable would be `60`. The `vu_reload_rolling_range` can always be shortened at a later period.

The Entitlement Analyzer does not support the ability to truncate the data to a rolling period like the Reload Analyzer does. The App Analyzer only collects minimal rolling data, and defaults to 90 days back. As this rolling data is aggregated daily, it is not nearly as large as the Reload Analyzer and can be left at the default setting without issue.

d. Reload the application manually from the load script and ensure that it reloads without issue.
e. Navigate to your third-party storage provider and confirm that the QVDs have been written.

6. Put each application on a reload schedule if using Qlik Cloud for scheduling. The suggested cadence is an hourly rate. For the "Parent" monitoring applications, you will want their reloads to follow the completion of the "Child" applications, as they rely on their QVDs. For those, it is recommended that they are scheduled hourly but skewed from the children by 30 minutes. These applications however reload in a minute or two on less active tenants, so this padding can be made tighter to accommodate if desired.

"Parent" Tenant Monitoring Application Configuration

1. Each of the monitoring applications must be configured on the "Parent" tenant as well, but this time set to "Parent" mode. Follow the same instructions for a single "Child" tenant above, however, when at step 5a, toggle `vu_is_parent_app` to `1`.

2. In addition, as per the instructions in step 6 above state, the reload schedule should be hourly but skewed by 30 minutes (or potentially less) from the children so that the "Parent" monitoring apps reload after the "Child" monitoring apps.

The Entitlement Analyzer does not support analysis in "Parent" mode, while the App Analyzer and Reload Analyzer do. This is because OEM tenants leverage a single license across all tenants, and the Entitlement Analyzer does not display data in that manner, and therefore can show incorrect measurements. The OEM Dashboard provides this information.

OEM Dashboard Configuration

1. Download the *OEM Dashboard* application from Qlik Community.

2. Import the application into the desired Shared space on the "Parent" tenant. This application can coexist in the same space with the other "Parent" monitoring applications.

3. Confirm that all "Child" applications and subsequently all "Parent" apps have been reloaded.

4. Navigate to the load script.

5. On the **** Configuration ****** tab, find the `vu_qlik_meta_collection_parent_dir` variable and give it the same value as the other “Parent” applications (being cognizant of the relative path and if it’s collocated in the same space as those apps).
6. Confirm that all “Child” apps and subsequently all “Parent” apps have been reloaded.

7. Reload the application and confirm that there are no errors.

8. Place the application on a reload schedule if using Qlik Cloud for scheduling. The suggested cadence is an hourly rate; however, this application needs to reload after the “Parent” applications reload. So just as the “Parent” app is skewed from the “Child” apps by 30 minutes, the OEM Dashboard can be skewed from the “Parent” apps by 15 minutes.

   For example:
   - “Child” applications at 15 minutes after the hour
   - “Parent” applications at 45 minutes after the hour
   - *OEM Dashboard* on the hour

Optional Add-On: Console Settings Collector

In addition to the required monitoring applications, there is another application that the *OEM Dashboard* supports as a source called the *Console Settings Collector*. This application loads most of the setting values from the **Console → Settings** section within the Management Console of each "Child" tenant. This data is valuable to track what features are enabled across each "Child" tenant (such as *Automations, ODAG, Data Alerts,* etc.). Because many of the required API endpoints to fetch this data are private (and thereby unsupported), this application is considered optional.

Console Settings Collector Configuration

1. Download the *Console Settings Collector* from the *OEM Dashboard* page on Qlik Community.
2. This application follows the same setup required as the other monitoring applications for both "Child" and "Parent" configurations, with the following exceptions:
    a. All variables are found on the **** Configuration **** tab in the load script, including those that configure settings for multitenancy, as this application only applies to multitenant scenarios.
    b. In line with the above, the `vu_multi_tenant_enabled` variable does not exist, as it is irrelevant.
3. Follow the same instructions from the "Child" configuration section and the "Parent" configuration section for this application, taking note of the exceptions above.
4. Once configured and both the "Child" and "Parent" applications have been reloaded, navigate to the **Load Script** of the *OEM Dashboard*.
5. Navigate to the ****Optional Configuration**** tab and find the variable `vConsoleSettingsCollector`. Set this variable value to `1`.

    Example:
    ```
    SET vConsoleSettingsCollector = 1;
    ```
6. Reload the *OEM Dashboard*.
7. Validate that the data from the *Console Settings Collector* has been loaded by navigating to the *Dashboard* sheet, and then selecting the *Features Across Tenants* tab within the container.

Customizing Thresholds

The *OEM Dashboard* provides many out-of-the-box variable thresholds that can be adjusted in the application's load script on the *** Optional Configuration *** tab. Many of these thresholds are not only used for dynamic coloring within the application but are also used for calculating expressions. These expressions can then be used to set data alerts as well.

Screenshot (*** Optional Configuration *** tab)

```
//=========
// Whether or not to include the Console Settings Collector.
// This app provides data supporting which features/capabilities
// are enabled on each tenant. It is defaulted to off, as many
// of the underlying APIs it uses are private/experimental.
SET vConsoleSettingsCollector   = 0; // 1 is on, 0 is off
//=========

//=========
// How fresh should the data be?
SET vDataStaleAsOfHoursAgo  = 6;
SET vDataStaleColor         = 'ARGB(255, 231, 76, 60)';
//=========

//=========
// Gradient color thresholds for reload failures per month
SET vReloadFailureMonthlyThresh = 10;
//=========

//=========
// Gradient color threshold for reload duration
LET vReloadDurationThresh           = .125; // 3 hrs (percent of day)
SET vReloadDurationThreshPercent    = .6;   // 60%
//=========

//=========
// Gradient color threshold for session duration
LET vSessionDurationThresh          = .00347;   // 5 minutes (percent of day)
SET vSessionDurationThreshPercent   = .6;       // 50%
//=========

//=========
// If you are pulling an attribute from the apps description,
// such as a version number like {v1.0.0}, enter the match pattern
// start and end here as you would in the TextBetween() function
SET vDescriptionAttributePatternMatchStart  = '{';  // example '{'  -- if none, leave ''
SET vDescriptionAttributePatternMatchEnd    = '}';  // example '}' -- if none, leave ''
//=========

//=========
// Entitlement thresholds (in percents, e.g., .6)
// Colors must be in Hex (no #)
SET vProLowThresh           = .6;
SET vProHighThresh          = .8;
SET vAnalyzerLowThresh      = .6;
SET vAnalyzerHighThresh     = .8;
SET vCapLowThresh           = .6;
SET vCapHighThresh          = .8;
SET vOverageLowThresh       = .6;
SET vOverageHighThresh      = .8;
SET vBelowLowThreshColor    = '006580';     // blue
SET vBelowHighThreshColor   = 'FFC72A';     // topaz
SET vAboveHighThreshColor   = 'E7004C';     // red
//=========
```

“Parent” Tenant Monitoring

The “Parent” tenant can be monitored in two ways:

1. The monitoring applications can be additionally installed on the “Parent” tenant in their default state (the `vu_multi_tenant_enabled` set to `0`) and monitored as a single tenant, as they would be used traditionally for a Direct customer.
2. The monitoring applications can be installed on the “Parent” tenant in “Child” mode, which will add them to all parent apps, and subsequently, the *OEM Dashboard*. Ensure that you create a bookmark or have a method of easily filtering this tenant out of selections when looking to analyze “Child” tenants.

Frequently Asked Questions

1. *I'm reloading the OEM Dashboard, but I'm not seeing the data update as I've expected after making a change in one of the "Child" tenants.*
    a. The data in the *OEM Dashboard* depends on two events happening upstream (in order):
        1. The relevant "Child" monitoring application has successfully completed a reload.
        2. The relevant "Parent" monitoring application has successfully completed a reload.
    b. This means that you will either need to wait for the full schedule to complete, or trigger (manually or programmatically) each of the relevant applications to reload in that order *before* reloading the *OEM Dashboard*.

2. *I have accidentally set the "Parent" tenant as a "Child" in one of the monitoring applications. How do I fix this? (Though, this can sometimes be desired)*
    a. Fetch the **TenantID** for the Parent tenant. If you do not know it offhand, you can get it at `/api/v1/tenants`.
    b. Navigate to your third-party storage provider and locate to the `QlikMetaCollection/Tenants` directory.
    c. Find the subdirectory that matches the **TenantID** and delete that entire directory.
    d. Reload each "Parent" monitoring application.
    e. Reload the *OEM Dashboard*.
3. *I have accidentally set a "Child" tenant as a "Parent" in one of the monitoring applications. What do I do?*
    a. Setting the "Child" application as a "Parent" will only provide another "Parent" view. The application will load all data from the children, perform any transformations required, and then store the consolidated data back down to QVDs. As this is all that the "Parent" app does, this is nothing to correct outside of flipping it back to "Child" mode.

4. *In the OEM Dashboard, I am trying to build my own visualizations that analyze professional and analyzer entitlement assignments as well as capacity usage, and the numbers do not appear to be correct. Why might they be incorrect?*
    a. OEM tenants leverage a single license, so many of the master measures refer to the total number of entitlements purchased and the numbers of each consumed across all tenants (from `api/v1/licenses/overview`, which has the same response across all tenants). These master measures are tagged with `global`. Other master measures sum up the total users assigned by entitlement and calculate on a per-tenant basis. These measures are tagged with `by_tenant`. The default visualizations ensure that the appropriate master measures are used with their corresponding dimensions. For example, the “Tenant ID” and “Tenant Name” dimensions should not be used with the master measures tagged with `global`, as the same numbers will appear for every tenant. In summary, some master measures refer to the number of assigned users from the license itself, while others calculate the number from the assignments on a per-tenant basis. If you are unsure which to use, leverage the default visualizations.
    b. It is also possible that the applications have not been reloaded since there was a change. Ensure that both the “Child” applications and “Parent” applications have been reloaded (in that order) before reloading the OEM Dashboard to confirm.
    c. Lastly, there are edge-cases where a user might be assigned an entitlement but has yet to login. This can happen when a user is *invited* to a tenant via the Qlik IdP. In this scenario, the license endpoint will show that the license has been consumed, but the calculated `by_tenant` tagged master measures might incorrectly report the total number, as those count distinct user IDs, which for these users have not yet been created.

Known Limitations

Entitlement Analyzer

The *Entitlement Analyzer* cannot be analyzed itself while it is in “Parent” mode. The application was initially designed for visualizing data from a single tenant with a single license, so many visualizations would report incorrect numbers as the master measures are not designed to support this use case. The OEM Dashboard is intended to be the visual layer upon this data for multiple tenants.

In addition, this application once reloaded begins collecting data at the current month and builds forward from there, incrementally. Harvesting historical data farther back than the first month is not supported by this application.

Reload Analyzer

The *Reload Analyzer* will not collect data if it is the first app ever reloaded on a tenant. If this is the case, simply reload it again to begin collecting data.

About Qlik

Qlik's vision is a data-literate world, where everyone can use data and analytics to improve decision-making and solve their most challenging problems. Qlik provides an end-to-end, real-time data integration and analytics cloud platform to close the gaps between data, insights, and action. By transforming data into active intelligence, businesses can drive better decisions, improve revenue and profitability, and optimize customer relationships. Qlik does business in more than 100 countries and serves over 50,000 customers around the world.

Qlik LEAD WITH DATA