QAD Adaptive Applications
QAD Adaptive ERP 2022.1

Implementation Guide

Adaptive UX

70-3349-2022.1-Rev2

QAD Adaptive Applications

Adaptive UX 2022.1

December 2022

QAD Inc.
100 Innovation Place
Santa Barbara, California 93108
Phone (805) 566-6000
https://www.qad.com

Table of Contents

QAD Adaptive UX Implementation Guide

This guide provides technical information for implementing various features of QAD Adaptive UX into an existing QAD Enterprise Edition environment.

Before proceeding with any installation or implementation tasks, be sure to read the release notes.

Product Name Changes

Starting in September 2019, the name for QAD's complete portfolio of products is QAD Adaptive Applications. Additionally, QAD Adaptive ERP is the new name for QAD's flagship ERP solution. QAD Adaptive ERP includes the functionality previously associated with QAD Cloud ERP and QAD Enterprise Applications - Enterprise Edition, plus the QAD Enterprise Platform and Adaptive UX, which resulted from the Channel Islands program. Going forward, the terms QAD Enterprise Applications, QAD Cloud ERP, and Channel Islands will be deprecated but will remain in previous documentation and training materials. QAD's intention is to—as soon as possible—eliminate the use of the deprecated terms going forward.

Adaptive UX Installation

For QAD Adaptive ERP with Adaptive UX, QAD performs and manages the installation. For on-premise installations of Adaptive UX, see the *QAD Adaptive ERP On-Premise Installation Guide*, available for early adopters in the QAD Document Library.

Recommended Web Browsers

The QAD Web UI is only supported on current versions of Chrome and Safari web browsers. Although other web browsers can be used, you may experience differing levels of performance and user experience.

To make sure you have the latest security updates, set your Chrome or Safari browser to receive automatic updates from Google or Apple.

For tablet use, the user interface is only supported on iPad Pro (or newer equivalent) with the Safari web browser. Although other tablets can be used, you may experience differing levels of performance and user experience.

Security Configuration

QAD Enterprise Platform includes comprehensive security features. For more information about security, please see the *QAD Security Administration Guide*, located in the QAD Document Library.

Role Permissions Configuration

Roles are used to model the business processes that exist within a business enterprise. Roles determine the set of application resources that display for users when they access their permitted workspaces. In order to model your organization's business processes effectively, users need access to all the appropriate application resources required for them to perform their everyday business tasks. Adaptive UX arrives with a variety of predefined, pre-configured roles. These roles are provided as starting points for the roles you will create for your system and cannot be updated. You can review the default roles' associated permissions on the Role Menus and Role Permission screens and use these default settings as a guide when assigning permissions to new roles.

For more information about user and role configuration, read the *QAD Security Administration Guide*, *Users and Roles* (chapter 6), located in the QAD Document Library.

Quality Orders - Hiding the Specification and Specification Detail Fields

In Item Attributes and Quality Control (IAQ), the Specification and Specification Details fields can be hidden in grids on **Quality Orders** for specific roles. This is useful if you do not want a certain role to be influenced by the specification information when entering attribute values.

To hide the specification fields in every browse and grid in **Quality Orders**, you will need to use **Role Permissions** and the Display Specification for Results Entry options in **Inventory Control** and **Quality Control.** See the following table for more information:

Screen	What is Affected	Notes
Role Permissions	Quality Orders (Lot and Quality Type) • Quality Orders > Attributes panel (grid and details form) • Quality Orders > Test Records > Test Attributes panel (grid and details form)	The Specification and Specification Details fields are hidden for the specified roles that do not have read access.
Inventory Control > Display Specification for Results Entry field	Quality Orders (Lot Type) • Quality Orders > Attributes Details screen > Attributes browse	The values in the Specification and Specification Details fields are blank for all users, regardless of role.
Quality Control > Display Specification for Results Entry field	Quality Orders (Quality Type) • Quality Orders > Test Records Details screen > Test Attributes browse • Quality Orders > Attributes Details screen > Attributes browse	The values in the Specification and Specification Details fields are blank for all users, regardless of role.

Hiding the Specification Fields

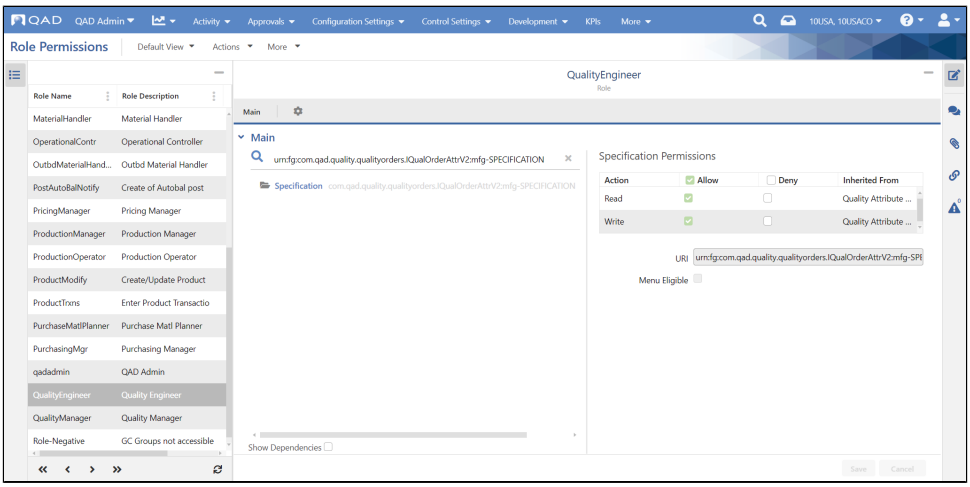
You can hide the Specification and Specification Details fields directly through **Role Permissions** or by using the role permissions functionality in the **Quality Orders** screen.

Hiding the Specification Fields From Role Permissions

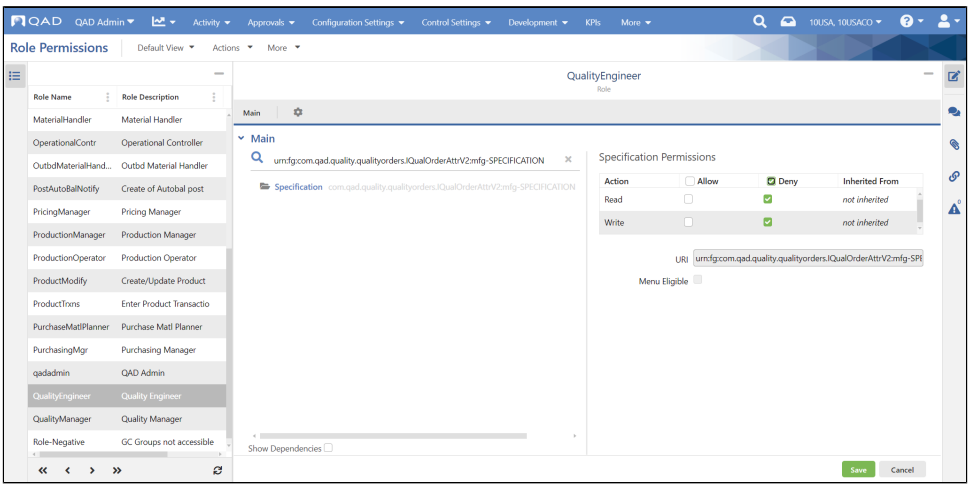
Follow these steps to use **Role Permissions** to hide the Specification and Specification Details fields:

1. In **Role Permissions**, select the role for which the specification fields will be hidden.
2. In the Search field, enter the URI for the components that will be affected:

Component /Screens	URI	Notes
Quality Orders > Attributes panel	urn:fg:com.qad.quality.qualityorders.IQualOrderAttrV2:mfg-SPECIFICATION	Hides the Specification and Specification Details fields in the Attributes panel (grid and details form) for quality orders (lot and quality type).
Quality Orders > Test Records > Test Attributes panel	urn:fg:com.qad.quality.qualityorders.IQualOrderTestAttr:mfg-SPECIFICATION	Hides the Specification and Specification Details fields in the Test Attributes panel (grid and details form) for quality orders (quality type).



3. To hide both fields, select Deny for Read and Write for the two URIs listed in the table.



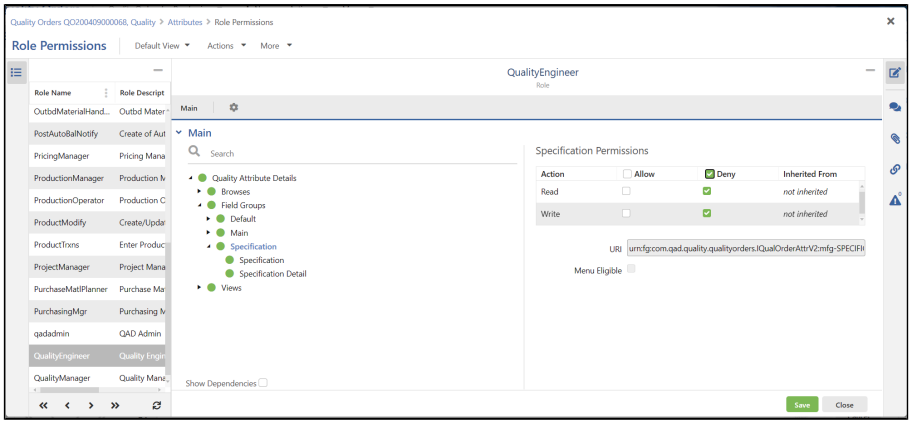
4. Select Save. The specification fields will now be hidden in the **Quality Orders** > Attributes panel and **Quality Orders** > Test Records > Test Attributes panel.

Hiding the Specification Fields From Quality Orders

Alternatively, you can hide the the Specification and Specification Details fields by navigating through the **Quality Orders** s creen.

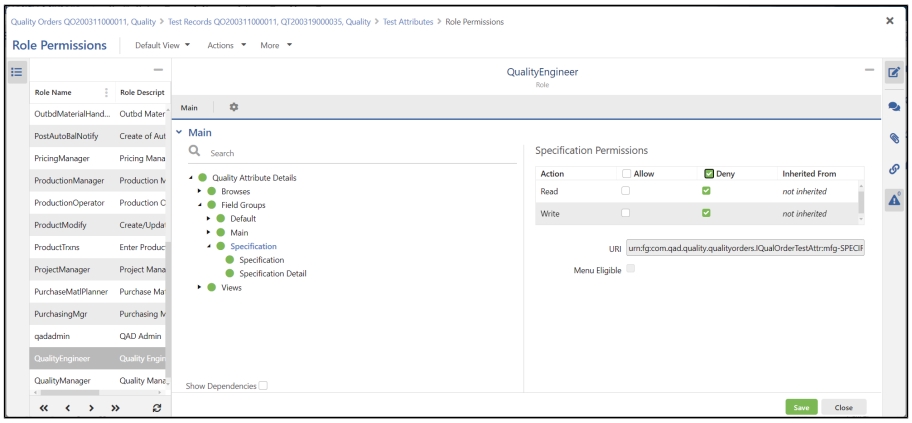
Follow these steps to hide the specification fields:

1. Hide the specification fields in the Attributes panel for quality orders (lot and quality type):
    a. In **Quality Orders**, select a quality order (lot or quality type) and open the Attribute details screen.
    b. From the More drop-down menu, select Permissions.
    c. In the Role Permissions screen, select the desired role.
    d. Then select Quality Attribute Details > Field Groups > Specification. The Specification and Specification Details fields are grouped together under the Specification field group. To hide both fields, select the Specification field group and then select Deny.



e. Select Save.
f. For the specified role, the Specification and Specification Details fields will now be hidden in the Quality Orders > Attributes panel.

2. Hide the specification fields in the Test Attributes panel for quality order (quality type):
   a. In **Quality Orders**, select a quality order (quality type) and open the Test Records > Test Attributes details screen.
   b. From the More drop-down menu, select Permissions.
   c. In the Role Permissions screen, select the desired role.
   d. Then select Quality Attribute Details > Field Groups > Specification. The Specification and Specification Details fields are grouped together under the Specification field group. To hide both fields, select the Specification field group and then select Deny.



e. Select Save.
f. For the specified role, the Specification and Specification Details fields will now be hidden in the Quality Orders > Test Records > Test Attributes panel.

Browse Performance Controls

Adaptive UX uses trusted queries to protect the system from long-running queries, which can adversely affect server performance and degrade the user experience for everyone in the system. Normal business component browse queries time out after 30 seconds by default. Trusted queries can be set to a longer period of time to ensure an individual browse loads successfully. System administrators can update these settings using the properties described on this page.

Update System Timeout for Normal BC Browses

To increase or decrease the timeout allowance for normal business component browses:

1. Open the `/build/config/configuration.properties` file.
2. Enter a value greater or less than 30000 milliseconds (30 seconds) for the setting `qad-qracore.bcbrowse.acceptableQueryTimeMillis=`
3. Run `yab reconfigure`.

Configure Timeout for an Individual BC Browse

The following property sets timeouts for individual BC browses:

`qad-qracore.bcbrowse.timeouts=`

The property is comprised of a comma-separated list of timeouts for particular BC browses with the format: `browseURI [timeout]`, where the timeout is specified in milliseconds.

Browse-specific timeouts override the default timeout. If you set this property for an individual browse to a value less than the system default, the browse will time out at the lower value regardless of the system setting.

To increase the timeout value for a browse:

1. Open the `/build/config/configuration.properties` file.
2. Define the timeout value for the browse using the `qad-qracore.bcbrowse.timeouts` setting.
   For example, to increase the number of seconds the Safety browse can load to 60 seconds, you would enter: `qad-qracore.bcbrowse.timeouts=urn:browse:bebrowse:com.extensions.qadextensions.safety[60000]`
3. Run `yab reconfigure`.

Custom Browses and Drill Downs

Browse Naming

As an administrator, you can define custom browses using Browse Maintenance in the QAD .NET UI. These browses are available in the QAD Web UI when the qra-sync YAB command (yab qra-sync) is run.
When defining custom browses, the labels for menu items on the QAD .NET UI are defined in Menu System Maintenance (in the Label field), while the labels for menu items in the QAD Web UI are defined in Browse Maintenance (in the Description Term field).

When defining a custom browse, be sure that the name of the browse (the label) is unique to avoid user confusion. Note that the browse naming convention on the QAD .NET UI is to have the name of the browse end with "Browse," while on the QAD Web UI, the menu item type is indicated by the menu type icon. Do not include "Browse" at the end of the name.

To ensure consistency, the Menu System Maintenance Label field setting should match the Browse Maintenance Description Term field, but do not include "Browse" at the end of the QAD Web UI's string.

Drill-Down Link Definitions

As an administrator, you can define drill-down links on the QAD .NET UI using Drill-Down/Lookup Maintenance.

For the drill-down links to be included on a QAD Web UI hybrid view, you first need to identify the browse used by the hybrid view of interest. To identify the browse used by a hybrid view, locate the browse on Role Permission Maintain's Secured Resources tab and get the browse identifier. For example, the browse used with the Sales Orders hybrid view is identified as "so803.p".

With the browse identifier, you can then define the drill-down links for it using Drill-Down/Lookup Maintenance. Note that in Drill-Down/Lookup Maintenance, in the Calling Procedure field, the browse identifier includes "br". For example, for "so803.p", in the Calling Procedure field, you enter "sobr803.p". You can then specify the Procedure to Execute.

Once the link has been defined in Drill-Down/Lookup Maintenance, the change is included on the QAD Web UI after an administrator runs the following YAB commands:

- Update resource dependencies: `yab qra-resource-dependency-update`
- Restart Tomcat: `yab tomcat-webui-stop tomcat-webui-start`

Action Centers

This section describes the architecture and components supporting the Action Centers, along with instructions for installing, configuring, and troubleshooting them.

It does not comprehensively cover installation or administration, but highlights points specific to the Action Centers and refers to other guides as needed.

Logi Analytics Technical Overview

The Action Centers are implemented using analytics tools and frameworks provided by **Logi Analytics**, a QAD partner owned by **insightsoftware**. Logi software supports the creation, maintenance, and display of the Action Centers and their contained visuals, which can be completed by trained users as a self-service activity not requiring additional software development. Visuals (charts, tables, gauges) can be created and modified in an iterative manner, shared with other users and QAD roles through a common gallery, and included in Action Centers by QAD users in roles with the correct permissions. Logi components have been embedded into the Web UI and integrated with the QAD data sources – mainly KPIs and their underlying browses.

Three Logi Frameworks

Depending on the AUX release, Logi provides one of three frameworks for the development and display of graphical analytics: **Logi Info, Logi Platform Services** (LogiPS), and **Composer**. **Logi Info** is the oldest framework, and was the only one used in Channel Islands before the September 2019 release. **Logi Platform Services** was introduced in the September 2019 release to replace Logi Info. **Composer** was introduced in the September 2022 release to replace Logi Platform Services, and is the only one available for customers installing the September 2022 release from scratch.

For releases prior to September 2021, Logi Info is provided, but is enabled for read-only use so that existing Action Centers provided by QAD or built by the users can still be displayed. For the September 2019 through March 2022 releases, all KPIs created in the KPI screen of the Web UI automatically use Logi Platform Services for creating and displaying visuals. The visuals for existing KPIs that were created using Logi Info can be displayed but not changed. Instead, the KPI must be copied to a new KPI that uses Logi Platform Services and the visuals re-created using the Logi Platform Services functionality.

> As of the September 2021 release, Logi Info is not supported and KPIs created using Logi Info can no longer be displayed.

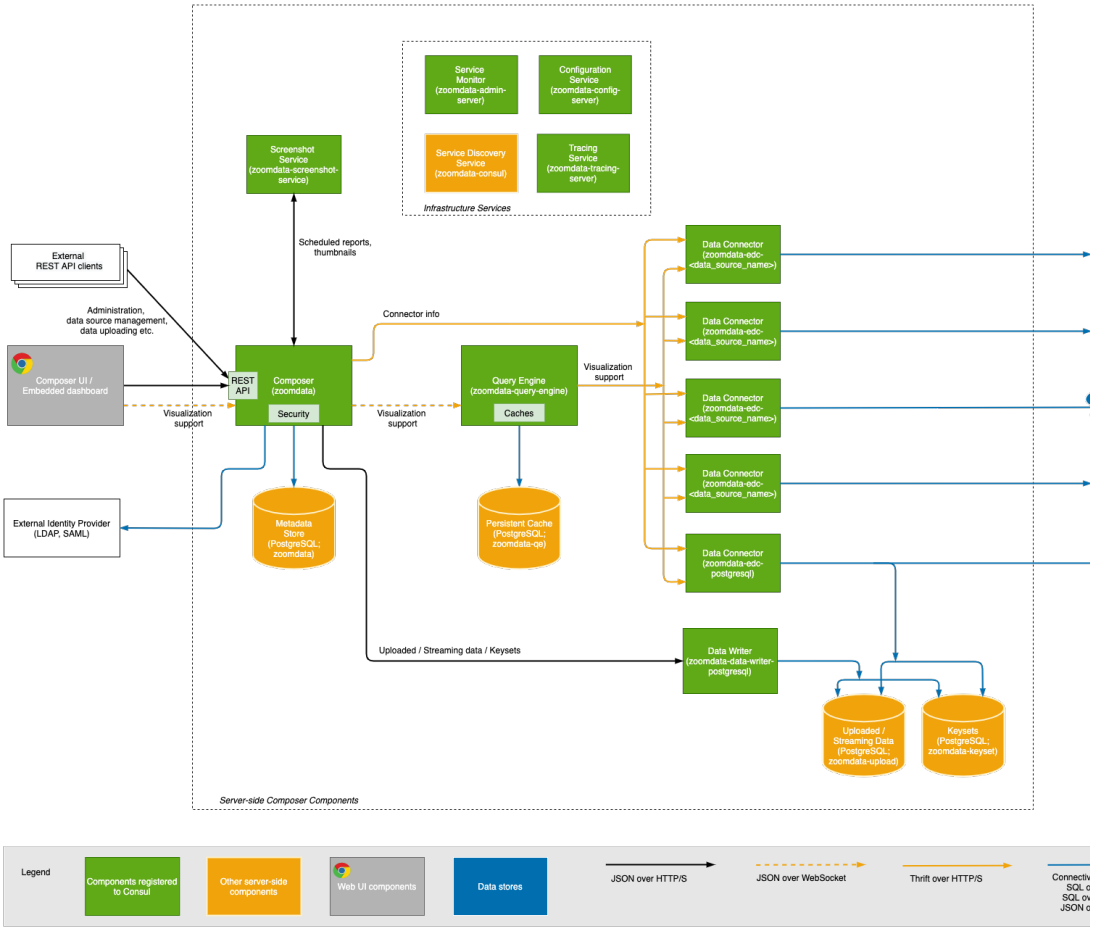
As of the September 2022 release, Composer is the primary analytics framework and the only one supported for new AUX installations. Customers upgrading from earlier releases have the option to continue using Logi Platform Services, although Composer is the default option. When upgrading an AUX environment to Composer, the customer's existing Action Centers and visuals are automatically converted for use with Composer, although some manual rework is still required. Once AUX has been installed with Composer, it is not possible to revert the environment back to Logi Platform Services. QAD plans to retire Logi Platform Services in a future AUX release.

> September 2022 environments installed from scratch or upgraded to use Composer cannot be reverted back to Logi Platform Services, which will be retired in a future AUX release.

The technical overview describes the high-level architecture of Composer, Logi Platform Services, and Logi Info.

Composer

Composer has a multi-tiered architecture built around a suite of microservices implemented in Java, using a PostgreSQL database to persist all configurations.



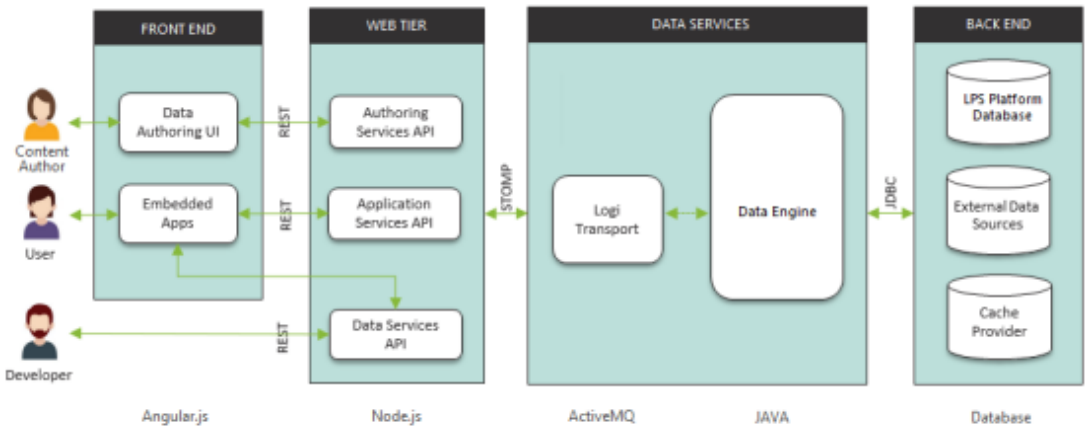
The web-based user interface supports the HTML5 standard for display and the WebSockets communications protocol, implemented using ReactJS and native JavaScript. The use of WebSockets provides faster transmission of analytics data to-from the browser client than approaches used in earlier Logi frameworks.

The services tier is made up of an extensible mesh of microservices, from which QAD has selected the ones needed to support current AUX Action Center capabilities: Query Engine, Configuration, Service Discovery. Other microservices may be added in future releases.

Composer offers a wide set of data connectors, also configured as separate microservices, to access different databases and other data sources. For AUX, two data connectors are used: PostgreSQL (required) and SparkSQL. The SparkSQL data connector reads analytics data directly from the Apache Spark ThriftServer through SQL queries.

Logi Platform Services

Logi Platform Services (LogiPS) uses a multi-tiered, service-oriented architecture to deliver rich visualizations and dashboards (Action Centers) in a more compact and modern form than other products.



The four components of the LogiPS architecture include the Front End, Web Tier, Data Services, and Back End.

LogiPS provides system administrators and content creators with an easily secured and configured system that can be embedded in web sites and applications without the use of iFrames. Visualizations, dashboards, and reports are created and stored in libraries, and the required code for embedding them is generated on request.

LogiPS uses an advanced data retrieval technology based on the *Dataview*. A Dataview is defined by a JSON document that specifies connection information, query details, and data enrichment options. This Dataview Definition (DVD) is stored in a system database and executed at runtime.

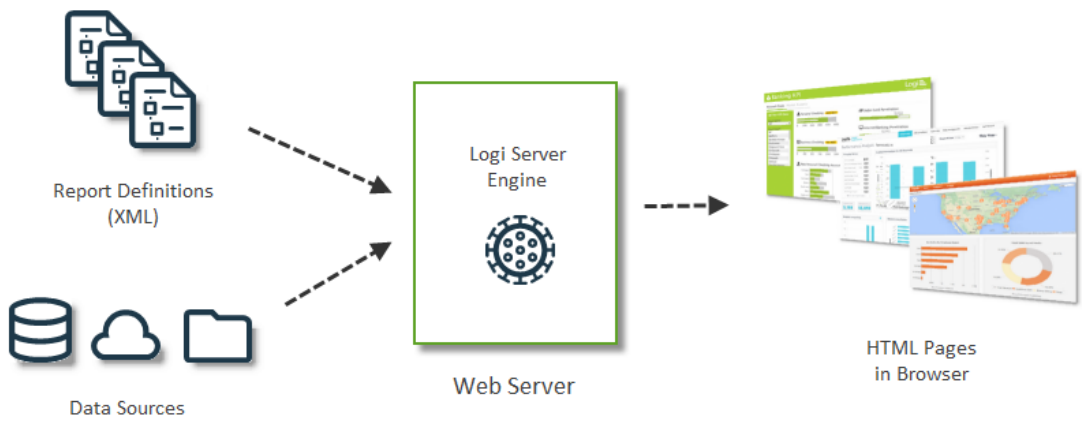
When executed at the server, a Dataview retrieves data, caches it in a self-tuning columnar data store, and makes it available for use with visualizations. The benefits of this include:

- Decoupling of the data and the visualizations
- Dataviews can be shared among multiple users
- Dataview changes manifest themselves immediately everywhere the Dataview is used
- Dataviews provide excellent performance for very large datasets (250M+ rows)

For Action Centers, the data is retrieved from the Query Service with SQL queries, as referenced in later sections of this document.

Logi Info

Logi Info is a framework for developing and displaying analytical data embedded inside a host application, in this case QAD Adaptive Applications. The resulting functionality is packaged and deployed as a web application, rendering visualizations and serving them to the browser as HTML pages.



A Logi application consists of web page sources known as 'report definitions.' Some of these pages provide robust self-service features that allow end users to define their own charts and cross-tabulation grids interactively, given the source data and metadata. These visuals can then be published and added to user-defined dashboards (Action Centers).

Logi Info applications separate the development, data access, and presentation processes, as shown in the graphic.

1. *Report Definitions* are text files that contain the information describing report layout and contents, stored as XML documents. While it is possible to create and edit definitions with any text editor, Logi Studio provides an integrated development environment with tools and helpful wizards that do much of the coding for you, reducing development time and effort.
2. When a report page is requested by a user, the Logi Server Engine**,** on the web server, processes the report definition and accesses whatever *data sources* are required. A wide variety of data sources are supported and data caching is used to speed up performance.
3. The Logi Server Engine formats the retrieved data and presentation details based on the definition and accompanying style sheets, generates *HTML* and *JavaScript*, and returns the report page to the user's browser for viewing.

There are two main data sources accessed by Logi Info to populate the Action Centers.

1. *Browses*: The same kinds of browses that can be defined by end users and displayed from the standard menu can also be used to retrieve data for the Action Centers through APIs called by Logi Info. The data sets consumed by Logi Info consist of the records returned by the browses.
2. *Financials Report Writer (FRW)* KPIs: The Financials Report Writer allows financial users to define key performance indicators using the enterprise's financial data, chart of accounts, and reporting structure. This information is also provided to Logi Info through APIs.

Query Service Technical Overview

This section describes the high-level technical architecture of the Query Service, a component that returns the business data required to display the Action Centers. The Query Service was introduced in the September 2018 release of Channel Islands.

Key Concepts

Reasons for Change

Previous Action Center releases had performance limitations when summarizing and displaying large amounts of source data, principally the very large result sets returned by some browses. In particular, performance degraded rapidly as the number of records returned by a browse increased. To mitigate this problem, the source data that could be included in Action Centers was limited to 5,000 records or less. The performance limitations have several causes.

- The overhead of processing browse requests in Progress AppServer agents, which often take a long time to complete and have high CPU usage. AppServer-based solutions are therefore difficult to scale up for high-volume environments.
- The high memory usage and slow processing time required in the Action Center web application to load and display very large data sets. The Logi Info software powering the Action Centers is optimized for self-service visualization and rendering, not high-volume data grouping and aggregation.

To address these issues, the architecture supporting data queries, post-processing, and retrieval was implemented using a Query Service.

Query Service

The original browse data retrieval engine was supplemented and partially replaced by a new infrastructure known as the Query Service. The Query Service incorporates several concepts and third-party components to bring required source data into the Action Centers more quickly and in a more compact, usable form.

SQL-Based Retrieval of Business Component Browses

The Query Service supports the use of virtually all kinds of QAD browses as data sources for the Action Centers.

- MFG (operational) browses
- FIN (financial) browses
- BC (business component) browses

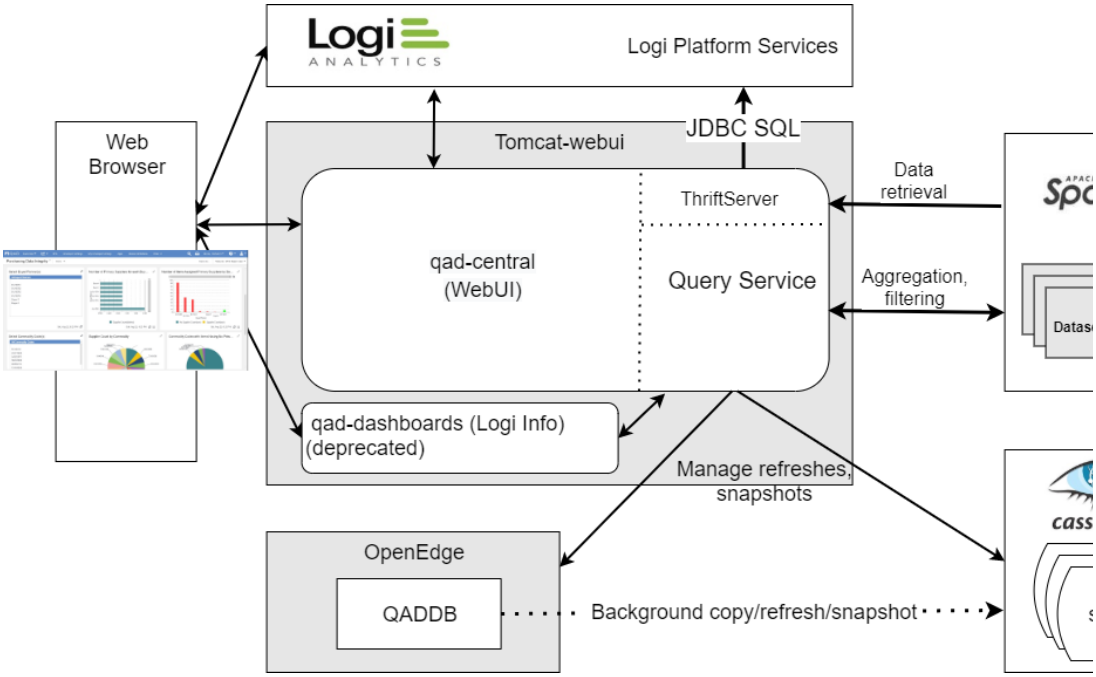
In the case of BC browses, infrastructure is provided to retrieve the OpenEdge source data through JDBC connections, rather than ABL code running in a Progress AppServer. This approach significantly reduces the CPU overhead and processing time to retrieve the browse data. It also leverages open SQL and JDBC standards, as opposed to the more closed and proprietary Progress AppServer architecture.

While current QAD-defined KPIs are based on MFG and FIN browses that rely on the older AppServer-based browse engine, in future releases this is expected to change as user-defined business components are more widely used and legacy browses are migrated to the newer infrastructure with SQL-based retrieval.

Data Lake

Data feeding the Action Centers is copied from its OpenEdge sources into a separate data lake repository built on Apache Cassandra. Cassandra stores the information in de-normalized form, where the relational data is joined and flattened before being written. Cassandra's columnar data structure is optimized for immutable (read-only) storage and fast retrieval, unlike relational databases such as OpenEdge, which are designed to support general-purpose CRUD operations.

For Action Centers, the data stored in the data lake is refreshed from its OpenEdge source overnight by default, when scheduled refresh is enabled both at the system level and for individual KPIs. As previously mentioned, the retrievals are processed using a combination of JDBC-based queries and Progress AppServer agents, depending on the browses needed. Refreshes may also be requested manually by end users for individual KPIs in Action Center panels.



Beginning with the September 2020 release, the data lake is also used to store historical snapshots of KPIs, for those KPIs that are defined as 'historical' instead of 'current.' This functionality allows KPI results from different points in time to be presented together for comparison and trend analysis purposes. The frequency of the snapshots and the total number allowed in the data lake are limited in order to manage disk space effectively, but are configurable by KPI. Unlike the scheduled refreshes of current KPI data mentioned previously, the historical snapshots are not re-createable from the OpenEdge sources. For historical KPI snapshots, the data lake is therefore the system of record.

In future releases, the data lake is planned to support other functions outside of Action Centers including historical archiving, reporting, and business intelligence. Its use will not be confined to Action Centers. However, supporting the Action Centers is its role in the current release.

In-Memory Post-Processing

Browse data stored in Cassandra is cached in memory using Apache Spark. Spark is a fast, scalable, in-memory data processing layer that can respond to queries in a flexible way, using any database fields as indexes. It also applies filtering, grouping, aggregation, cross-tabulation, and deduplication on demand much faster and more efficiently than could be done by OpenEdge ABL code running in a Progress AppServer.

At runtime when an Action Center is displayed, the data can be pre-grouped and pre-aggregated by Spark based on the definition of the KPIs associated with that Action Center. The data returned to the Action Center is much more compact as a result with fewer records, allowing the Action Centers to overcome the 5,000-record limit for source data prescribed in previous releases.

Optimized Logi Integration

Older Action Center releases leveraged the Logi Info framework from Logi Analytics as the basis for Action Center definition and display. The Logi-based Action Center artifacts are deployed as a separate web application, named qad-dashboards by default, inside the tomcat-webui server. This web app is tightly integrated with the primary QAD Web UI web application, with Logi screens embedded inside the Web UI window and menu. REST API calls passing between the two web apps are used to communicate user directives, metadata, and business data into the Logi environment.

Beginning with the September 2018 release, the Logi integration was enhanced to take advantage of the superior performance of the new Query Service. As end users design visuals and their data contents using the Action Center's self-service capabilities, metadata defining the group-by categories and numeric aggregations needed to support those visuals is automatically extracted from Logi and sent to the Query Service. The Query Service is then able to apply the maximal a mount of grouping and pre-aggregation that Logi can consume in order to render the requested charts. In summary, the Logi integration optimizes its request to obtain exactly the required data in pre-processed form, with the result set made much smaller through Query Service grouping.

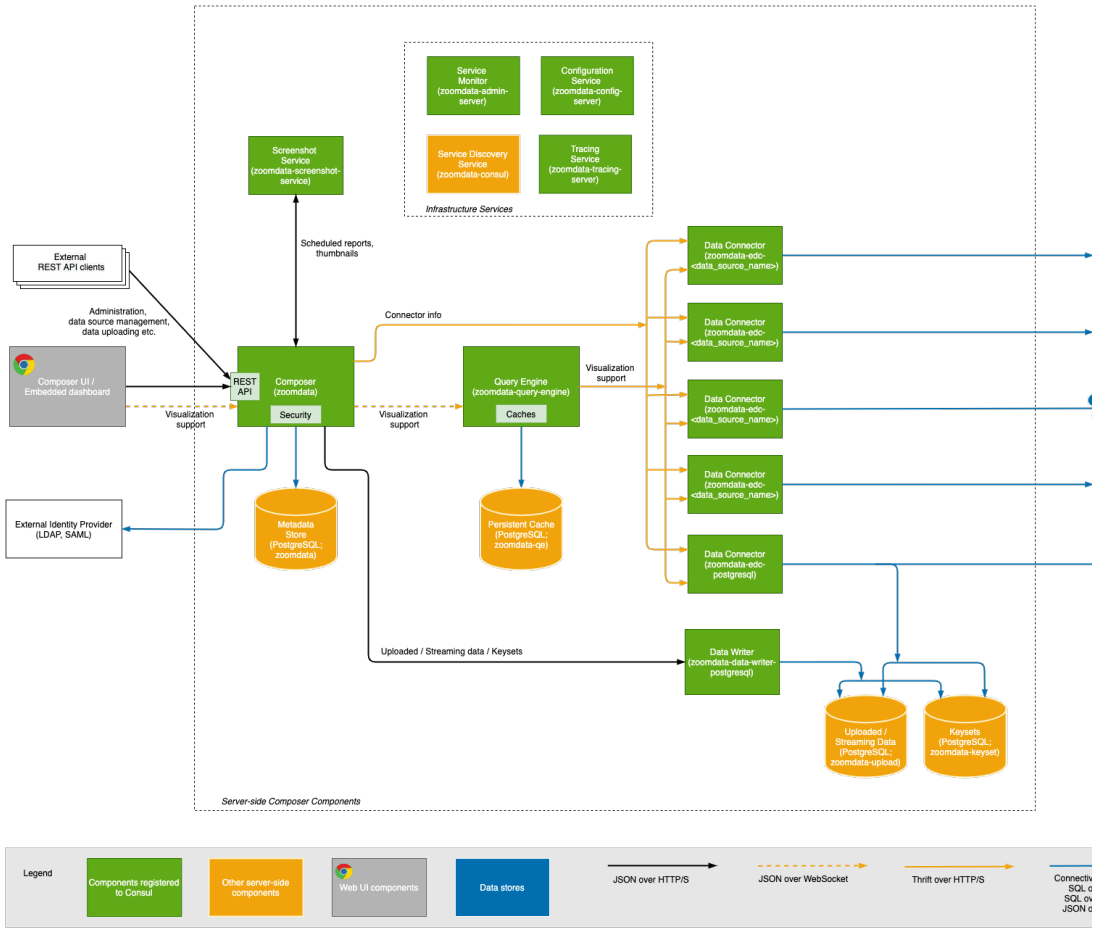
Beginning with the September 2019 release and the introduction of Logi Platform Services (LogiPS), the Logi integration was re-engineered again to allow Logi to retrieve the Query Service data using SQL through a JDBC connection, which is supported inside the Query Service by the Spark ThriftServer. This approach allows Logi to retrieve the data as though from a relational database, which better leverages the built-in capabilities of LogiPS to group and aggregate the data automatically to suit the needs of each visual.

Action Center Key Components

The Key Components section describes the major third-party components that are critical to the Action Center architecture.

Logi Composer

Logi Composer replaces Logi Platform Services as the analytics framework used to maintain and display Action Centers. Unlike previous Logi frameworks, it is built as a set of loosely coupled microservices that can be selectively installed depending on the capabilities required. These microservices are implemented in Java using Spring Boot, and managed in a single service mesh. Many of its components use the internal name "zoomdata".



AUX uses the following Composer microservices.

Query Service

The Composer Query Engine is responsible for processing and dispatching queries to retrieve Action Center data, sitting between the WebUI and the data connectors that directly access the data sources. It has the following primary tasks.

1. Deconstructs and converts user query requests into distributed execution plans.
2. Optimizes the execution plans based on data platform capabilities, in-memory cached results, and the Query Engine capabilities.
3. Communicates with Composer data connectors to execute push-down queries. In the case of AUX, the SparkSQL data connector is used.
4. Uses in-memory processing to combine, append, or manipulate one or more data sets to produce only the values needed to fulfill the request.

Data Connectors

Composer offers a wide set of data connectors, each configured as a separate microservice, to access the sources of analytics data. While many kinds of databases are supported, AUX uses the **SparkSQL data connector** to read Action Center data from the Apache Spark ThriftServer, where it is cached in memory. The **PostgreSQL data connector** is also required, in order to access the Composer configurations stored in its native PostgreSQL database. In addition, it is possible to create custom data connectors for other data sources.

Service Discovery

The Service Discovery microservice helps manage the mesh of microservices by integrating Composer with Consul, an open source tool that supports secure access to and communications across microservices. It provides:

- External configuration capabilities for Composer via a key/value store.
- A service discovery client backed by the Consul service registry.

Other Microservices

Various other microservices are available in Composer but not yet used by AUX. It is possible that some of these may be included in future AUX releases.

Configuration

The Composer Configuration microservice is packaged with the Spring Cloud Configuration server, which allows Composer to easily integrate with its Spring-based microservices. It provides the mechanism by which Composer property settings can be maintained using the Service Monitor, another Composer microservice. The property settings are persisted in a supported PostgreSQL data store or in a GitHub repository.

Service Monitor

The Service Monitor microservice provides administrators with real-time views of microservice health, configuration, and logs. The Service Monitor is not installed as part of a default Composer installation, so it must be installed and configured before it can be used. It allows system administrators to do the following tasks.

- Review all Composer microservices and their log files.
- Produce a diagnostics bundle to send to Composer Support.
- Set the logging level and properties for each microservice.

Data Writer

Composer offers a multi-purpose Data Writer microservice that writes data to a relational database for an enriched analytic experience. Its current uses are to:

- Persist user-uploaded flat text or CSV files for reuse, performance, and functional improvements such as push-down processing and derived fields.
- Simplify landing or persisting streaming data into a high-performance data platform for subsequent analysis.
- Store user-generated keysets for “set analysis” to be used in single and multisource data environments.

Uploaded data is stored in the separate "zoomdata-upload" PostgreSQL DB that can follow the same backup/restore process as the main zoomdata database.

Screenshot

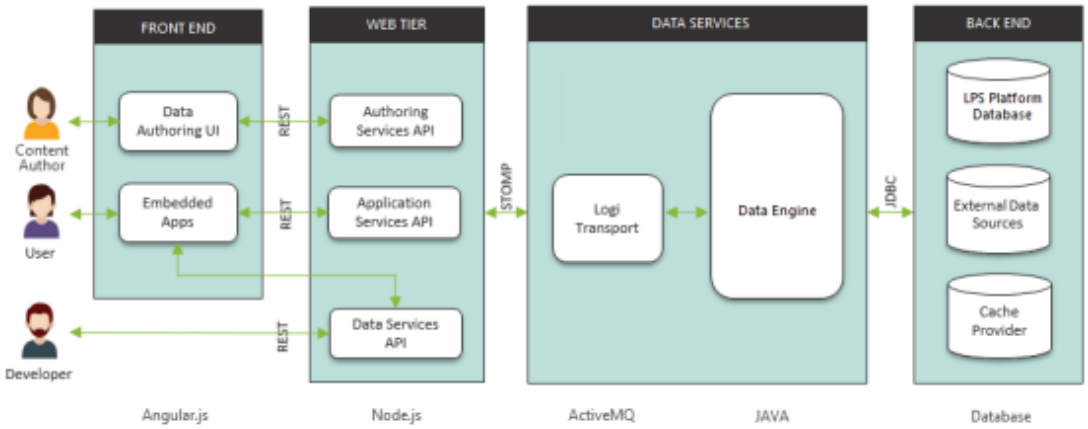
The Screenshot microservice allows users to view snapshots of saved dashboards.

Logi Platform Services

Logi COMPOSER

Starting with the September 2022 release, Logi Composer is the only option available to install in new AUX environments and the default option for existing AUX environments. While Logi Platform Services can still be used in older AUX environments that have been upgraded to September 2022, it is deprecated and will be retired in a future AUX release.

Logi Platform Services (LogiPS), a.k.a. Logi Composer, uses an n-tier, service-oriented architecture that layers presentation services, application services, and data services into distinct sets that can be deployed logically and physically at different tiers in an environment.



The four components of the LogiPS architecture include the front end, web tier, data service, and back end.

In a front end client, a developer embeds the desired Logi widgets into an HTML web page or application. Widgets are configurable, client-side components that are used to create visualizations. Multiple widgets can be embedded in a web page and they can communicate with one another. Content authors and users will use the client to interact with LogiPS. The complex widget used to build visuals is called a **Thinkspace** view.

The front end runs on Angular.js, which supports the model-view pattern on the client. Angular uses Restangular, a service that handles REST API requests, to communicate with Logi application and data services. Responses are delivered with JSON data.

The web or **application service** tier passes the HTTP requests to the appropriate handler on the server. It uses Node.js, an embedded, run-time environment for server-side web applications that handles all server-side processing at the application level. Node.js uses the Express framework, which is a lightweight web server with full REST support.

The LogiPS **data service** processes and queues requests from the application service using the ActiveMQ message broker, communicating via the STOMP protocol. ActiveMQ passes the request to the Logi Transport service, which interprets it and makes a request to the Logi Data Engine for retrieval of the requested data.

LogiPS uses the back end via JDBC to retrieve and process the appropriate Dataview (stored in the Platform Database), which in turn retrieves the appropriate data from an external data source or cache.

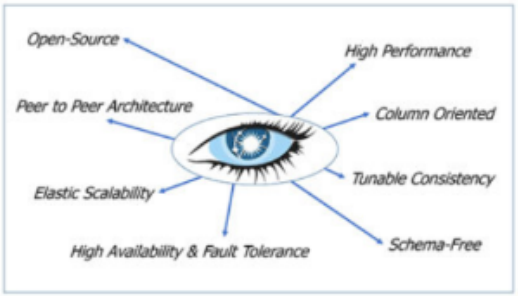
The server-side components are run within two separate processes, which are started and stopped separately.

- **Data Service**: Includes the data services tier, containing the data engine and controlling the LogiPS Platform Database (PDB).
- **Application Service**: Includes the web tier and the REST APIs called from clients to maintain the contents of the PDB.

The self-service capabilities of Logi Platform Services are enhanced to consume larger volumes of source data in a more performant way using the Query Service's other components, which are described in the following sections.

Starting with the September 2021 release of AUX, Logi Info and KPIs created using Logi Info are no longer supported.

Apache Cassandra



Not a Relational Database

Cassandra is not a relational database intended for transaction processing. Rather, it is a multi-node, horizontally scalable, columnar database with an embedded SQL-like language. It is used as a standard data lake repository by many of the largest companies in the world due to its speed, reliability, distributed architecture, and flexible data model.

In small- to medium-sized environments, Cassandra will be deployed on the same node as the core Enterprise Application Infrastructure. In larger environments, it may reside on a dedicated server as part of a larger data lake infrastructure. In very large, multi-site enterprise environments, Cassandra could be deployed as a decentralized cluster with multiple nodes.

While Cassandra is a columnar database, many of the key concepts with which application developers and DBAs are familiar also apply to Cassandra. There are tables, records, and fields. However, data is internally organized into columns rather than rows, unlike relational databases such as OpenEdge and Oracle. This makes Cassandra well suited to high-volume queries and analytics, where users often drill down by columns (for example, "Show me all the data for this item or this customer") rather than transactions, as in traditional ERP (for example, "Create an invoice for this customer").

Columnar databases support a flexible schema model and much improved performance for analytics, but do not support join operations or secondary indexes very well. In implementations that require the flexibility of user-defined joins and filters, these actions should be done in complementary frameworks such as Apache Spark (see below).

Some advantages of Cassandra are:

- High availability
  - Distributed table storage
  - Tunable consistency
  - Fault tolerance
- High performance
  - Low latency
  - Linear scalability
  - Table-specific tuning
- Data model flexibility
  - Dynamic schema controlled by queries
  - Management and monitoring via JMX and SQL queries
- Open source licensing

Terminology

The following basic terms are necessary to understand how Cassandra works.

- Cassandra is a distributed database running nodes in a *cluster*. The nodes communicate in a peer-to-peer, master-less fashion.
- Cassandra rows are stored in *tables*, where each table has a mandatory primary key.
- A *keyspace* groups tables as a logical entity, similar to a schema in relational databases. In the Query Service, all browse result sets are stored in the "browses" keyspace.
- Data is accessed via *CQL*, an SQL-like query language.
- Cassandra writes to a data log first, similar to the OpenEdge before-image file. It then writes to an in-memory cache inside a JVM heap called *memtables* before flushing to disk (SSTables).
- Cassandra housekeeps the data on disk, compacting it and discarding *tombstones*, which are markers placed inside obsoleted data to mark it for later physical deletion.

The following table cross-references Cassandra terms to similar concepts from OpenEdge.

Cassandra	OpenEdge
Cluster	N/A
Column	Field
CQL	ABL

Data Center	Replication Group
Data Log	Before-Image File
Flat Data	Relational Data
Key Cache	Buffer Pool
Keyspace	Database
Memtable	Buffer Memory
Node	Server
Row (single record)	Row (unit of replication)
SSTable	Record block
Table	Table
View	add another index

Compaction and Deletion

Cassandra collects all the versions of a row, and from them assembles the most up-to-date versions of that row. It then writes the new row versions to a new SSTable, leaving the old versions along with other rows that are ready for deletion in the old SSTables. As soon as all pending reads are completed, Cassandra deletes the old versions using markers called tombstones, which indicate that the data has been obsoleted.

After many deletes, the resulting tombstones can grow to consume a significant amount of disk space and slow Cassandra processing. However, Cassandra deletes tombstones automatically in its compaction runs, which are triggered every few minutes. By default, tombstones older than 10 days are deleted during compaction. While this time period can be configured if needed, in the case of Action Centers this should not be necessary. The browse data stored in Cassandra is only deleted during a scheduled or manual refresh, and the refresh causes the affected Cassandra tables to be truncated and re-populated. Hence, individual rows are not deleted and tombstones will not accumulate in the database.

Cassandra performs best with local low-latency SSD or SAS storage, which is also cheaper to provision than relatively high-latency network storage. It uses an efficient log–structured engine that converts updates into sequential I /O. Cassandra's storage engine does not read or rewrite existing data when processing updates, but only appends the updated data. This approach allows updates to be processed very fast. However, updates and deletes can be expensive and generate tombstones, which can affect query performance.

Core Tools

Several native Cassandra tools are useful for system administrators and DBAs. Basic database start, stop, remove, rebuild, and other functions are controlled through YAB and not listed here. To see a description of the YAB commands for managing Cassandra, use the command 'yab help cassandra-'.

- SSTable: Table utilities such as dump, print metadata, split table, list tables.
- nodetool: Comprehensive utility used to monitor and manage a cluster. This tool can also be started using the 'yab cassandra-default-nodetool' command.
- cqlsh: Command-line utility for connecting to a Cassandra database and executing CQL commands.
- cassandra-stress: Stress testing tool.

Some of these tools are implemented in Python, but Cassandra in general is entirely Java-based.

Memory Mapped Files

Cassandra uses memory-mapped files (mmap) internally. That is, the operating system's virtual memory is used to map a number of on-disk files into the Cassandra process address space. This uses virtual memory or address space, and is reported by O/S tools, such as top, accordingly. However, on 64-bit systems virtual address space is effectively unlimited, so it is seldom a concern.

A key point is that for a mmap'd file, there is never a need to retain the data in physical memory. Thus, whatever is in physical memory is there only as a temporary cache, in the same way that normal I/O will cause the kernel page cache to retain data that has been read or written.

The major difference between normal I/O and mmap is that in the mmap case, the memory is mapped to the process, thus affecting the virtual size as reported by top. The main argument for using mmap instead of standard I/O is that read actions only need to access memory; there is no page fault, therefore no kernel overhead to perform context switching.

CAP Theorem and Cassandra

The well-known CAP theorem states that it is impossible for a distributed computer system to simultaneously provide all three of the following guarantees.

- *Consistency*: All nodes see the same data at the same time.
- *Availability*: Every request receives a response about whether it succeeded or failed.
- *Partition* tolerance: The system continues to operate despite arbitrary message loss or component failure.

All databases can be categorized as CP, AP, or CA, based on which of the guarantees are supported. For example, OpenEdge and other relational databases are CA databases, while MongoDB and ElasticSearch are CP databases. Cassandra is an AP database. The advantages of an AP database are greatest in environments where short periods of data inconsistency are preferred over short periods of database unavailability.

Cassandra satisfies a weaker consistency requirement by adopting the BASE standard, which is a modified version of the ACID (Atomicity, Consistency, Isolation, Durability) properties satisfied by most relational databases.

- *Basically Available*: The system guarantees the availability of data in the sense that it will respond to any request. However, the response could be a failure to obtain the requested data, or a data set in an inconsistent or changing state.
- *Soft*: The state of the system is always “soft” in the sense that eventual consistency, described below, may cause changes in the system state at any given time.
- *Eventually Consistent*: The system will eventually become consistent once it stops receiving new data inputs. As long as the system is receiving inputs, it does not check the consistency of each transaction before it moves to the next transaction.

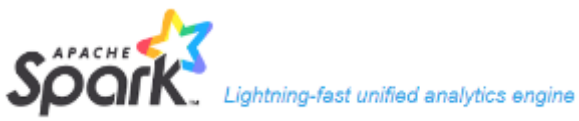
Full consistency has a negative effect on cost-effective horizontal scaling. If the database needs to check the consistency of every transaction continuously, a database with billions of transactions will incur a significant cost to perform all the checks. The idea of consistency is not practical in a large distributed database. It is the principle of eventual consistency that has allowed Google, Twitter, and Amazon, among others, to interact with millions of their global customers, keeping their systems available by supporting partition tolerance. Without the principle of eventual consistency, today's systems could not support the exponential rise of data volumes caused by cloud computing, social networking, and related trends.

Limitations

Cassandra is in some ways very restrictive compared to relational databases like OpenEdge, as summarized below.

- *Joins*: Cassandra does not allow joins, and is therefore not suitable for representing normalized data models. Join s must be implemented in a separate component such as Apache Spark, which is described in the following section. The data stored in Cassandra should be self-describing documents (for example, an invoice object in XML or JSON form) or de-normalized, flattened views designed for query purposes.
- *Transactions*: Cassandra supports only "lightweight" transactions, essentially only existence checks, without the ACID compliance common in relational databases.
- *Secondary Indexes*: Cassandra does not fully support secondary indexes, as it imposes heavy restrictions on which fields can be included in a secondary index.
- *Text Search*: Cassandra allows full text search with advanced features, but only on specific fields with a text search index.
- *Tombstones*: Cassandra does not delete and update data in real time like a relational database. In order to ensure good read performance, it simply marks the affected data with a tombstone. Future queries automatically skip over the tombstones. However, tombstones can build up quickly in tables with heavy delete/update activity, even to the point where there are more tombstones than actual data. In such cases serious performance problems can result, as reading tombstones can cause excessive latency, timeouts, and even exceptions. Tombst ones also consume disk storage unnecessarily.

Apache Spark

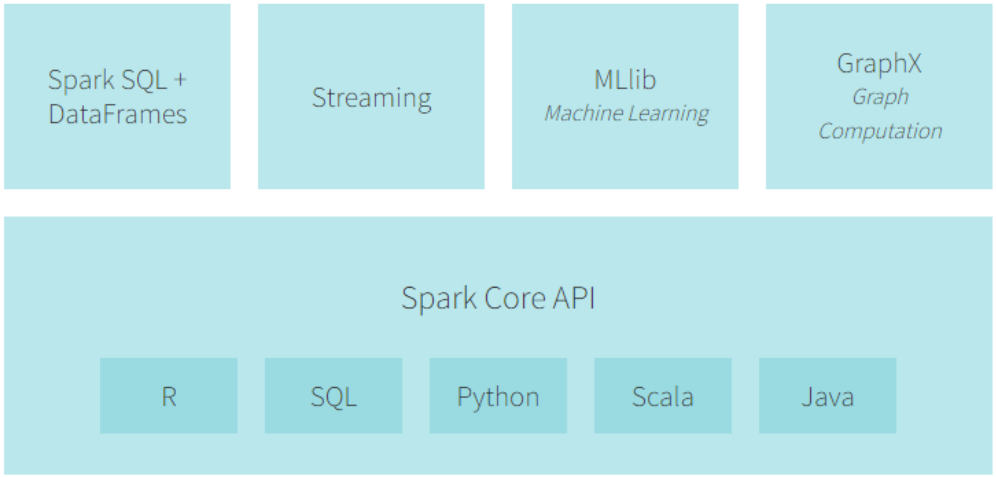


Apache Spark is a general purpose in-memory data processing engine and cluster computing framework that can perform Extract, Transform, and Load (ETL) operations, ad-hoc queries, machine learning, and graph processing on large volumes of data at rest (batch processing) or in motion (stream processing).

It supports native APIs for manipulating and querying data in the following programming languages: Scala, Java, Python, R. In addition, it provides libraries that allow the same data to be accessed through more specialized and advanced languages and protocols.

- *SQL*: A Spark module for structured data processing. It provides a programming abstraction called DataFrames to access data organized into named columns, like a relational table, and can act as a distributed SQL query engine.
- *Streaming*: Spark Streaming is a scalable fault-tolerant streaming system, receiving data streams and chopping them into batches. Spark then processes those batches and pushes out the result. Besides working directly with files and sockets, it integrates with a variety of popular data sources, including HDFS, Flume, Kafka, and Twitter.

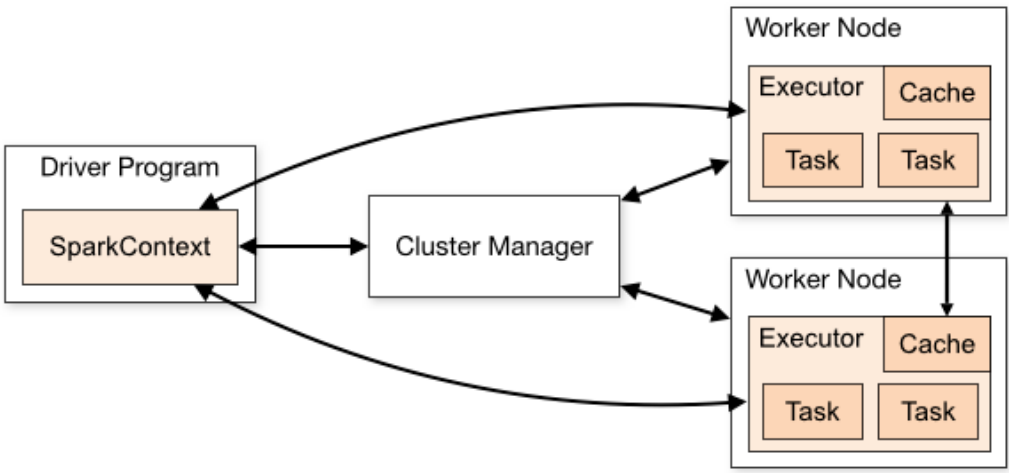
- *MLlib*: Built on top of Spark, MLlib is a scalable machine learning library that provides high-quality algorithms performing at high speeds. The library is usable in Java, Scala, and Python.
- *GraphX*: A graph computation engine built on top of Spark that enables users to interactively build, transform, and reason about graph structured data at scale. It comes with a library of common algorithms.



Architecture

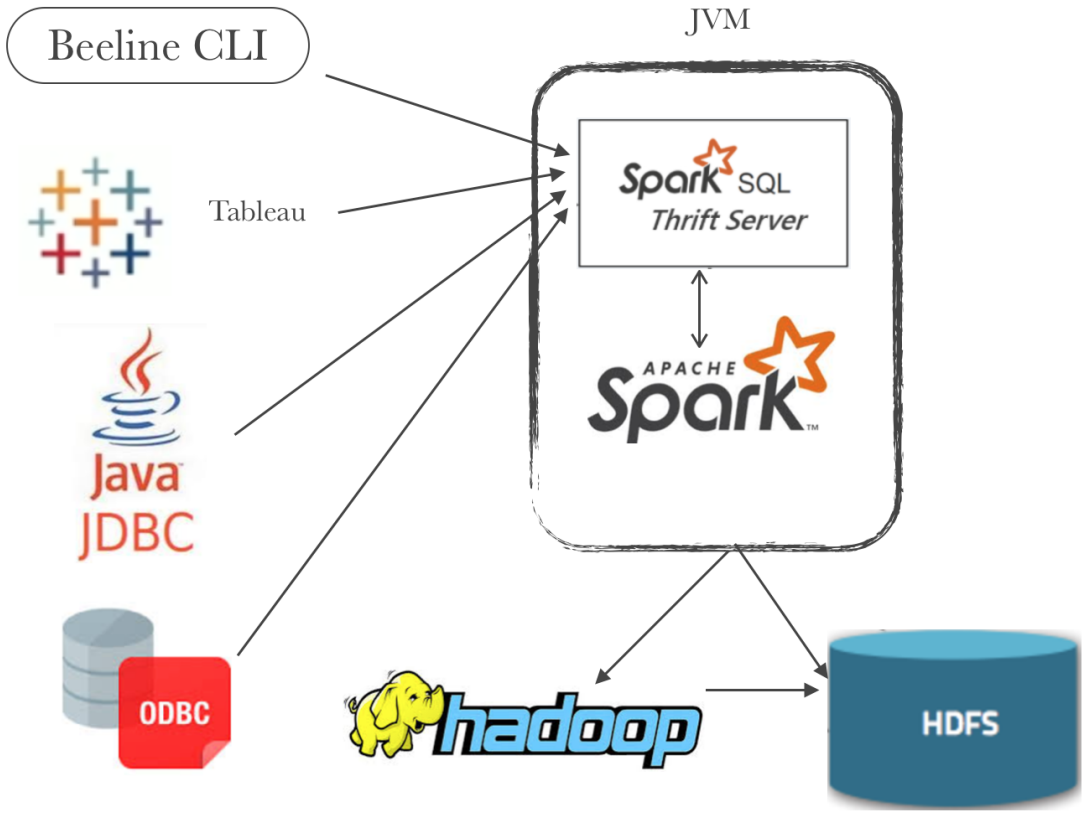
Spark applications run as independent sets of processes on a cluster, coordinated by the SparkContext object in a main program called the "driver" program. The Query Service has its own driver program and SparkContext instance, from which all the in-memory browse data can be accessed.

Spark can run standalone where all the necessary components are loaded at run time and jobs are executed. However, this method is (a) slow to instantiate, because of the need to load the components; and (b) difficult to manage, because each process has its own Java heap memory and resource requirements. In the Action Center context, the Query Service driver program connects to the Spark Cluster Manager, which accepts job requests and allocates resources across applications. Once connected, Spark acquires executors on nodes in the cluster, which are processes that run computations and store data for the application. Next, it sends the application code packaged in JAR or Python files to the executors. Finally, the SparkContext dispatches tasks to the executors to be run.



ThriftServer

The Query Service originally used Spark's native Java API to read and manipulate the browse data. With the introduction of Logi Platform Services in the Sep 2019 release, the data residing in the Query Service is retrieved directly by Logi as an SQL data source. This is accomplished by a component of Spark called the ThriftServer.

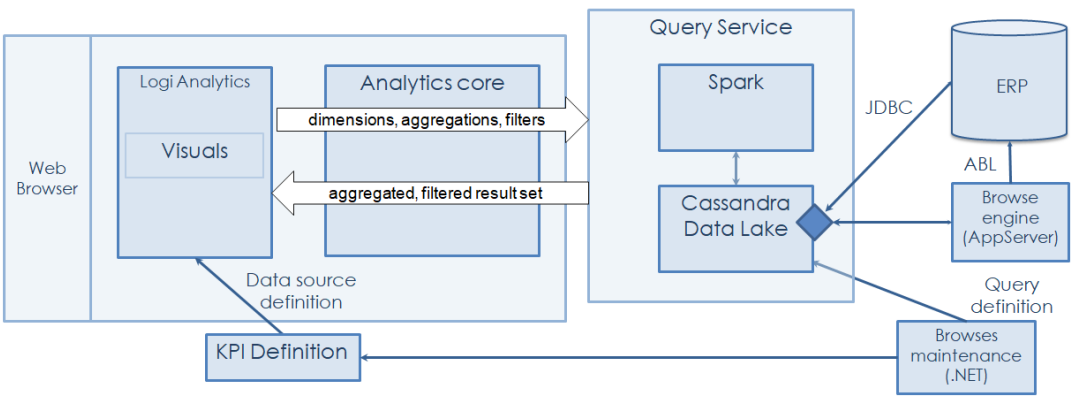


The ThriftServer is a server interface within Spark that enables remote clients to execute SQL queries and retrieve the results through a JDBC connection. In essence, it exposes the tables and views within Spark as an SQL database, supporting multi-client concurrency and authentication.

The ThriftServer is embedded inside the Query Service, which runs inside Tomcat. It requires the configuration of an additional port for Logi to connect through JDBC, but does not run inside a separate process.

Spark and Cassandra in the Query Service

In the Query Service, Spark and Cassandra are integrated to use the strengths and features of both to bring browse data to the Action Centers in a flexible and performant manner.



Spark serves as the in-memory cache where the data retrieved from browses is stored for on-demand retrieval. Before returning the browse results, it groups the data, pre-aggregates the numeric values within each group, and applies filters in order to return the minimum amount of detail needed to support Action Center display. Because grouping, aggregation, and filtering requirements can be changed by Action Center users at any time, Spark's combination of flexibility and speed is critical.

Cassandra serves as the data lake where the browse data is persisted, generally before it is needed. On a scheduled basis, browses that are configured for use in the Action Centers are refreshed from the operational OpenEdge database tables through one of two query mechanisms.

1. SQL with JDBC connections: Business Component browses are processed as SQL queries directly against the OpenEdge database. This approach is preferred, as it is significantly faster than the AppServer-based approach.
2. AppServer-based browse engine: Other browses are processed using the existing browse engine, which runs inside Progress AppServer agents and reads the OpenEdge data using ABL code.

End users can also request refreshes of a specific browse from the Action Centers, which causes the data to be refreshed in both Cassandra and Spark. In addition, beginning with the Sep 2020 release, historical snapshots of some KPIs are automatically created, stored in Cassandra, and cached in Spark. In future releases, browse results may be pushed into Cassandra more continuously as the source data is updated in Enterprise Applications through transaction processing activity.

The current KPI browse results extracted from the OpenEdge databases are stored in tables that reside in the Cassandra keyspace "browses." For most browses, there is a single table for each KPI and combination of browse and domain or browse and entity, depending on whether the browse was defined to access financial data, which are generally associated with financial entities, or operational data, which are generally associated with domains. The contents of these tables are then cached in Spark for online retrieval by the Action Centers.

This historical KPI snapshots are stored in table that reside in the Cassandra keyspace "historical_kpi". In this keyspace, there is a single table for each KPI snapshot. Because the number of historical snapshots per KPI can vary widely depending on KPI configuration, different historical KPIs can have different number of snapshot tables in the keyspace.

Action Center Installation

Review this section carefully before running the YAB installation, as it covers YAB properties that must be set and several procedures that must be completed prior to installation.

The Installation section describes useful details about how Action Centers and the related Logi and Query Service infrastructures are installed. It is not a comprehensive, step-by-step guide, as the installation process is largely automated through the use of the YAB tool. Action Centers are not installed in isolation, but as part of an overall release as documented in the *Adaptive UX On-Premise Installation Guide*. However, this section describes some installation steps in greater depth that are specific to Action Centers, referring to YAB command details and other guides as needed for context. It also covers important steps that must be completed before the automated installation is run.

This document assumes that the reader is familiar with basic Linux system administration and YAB. For more details on YAB, see the latest *QAD Configuration and Administration Guide* for YAB, available on the QAD Document Library.

System Requirements

Memory

Spark and Cassandra are fast because they do considerable amounts of in-memory processing. Logi Platform Services and Logi Info also require memory to render the visuals. The minimum memory requirement for running production systems with the Action Centers and Query Service is 16GB.

CPU

Systems running the Action Centers and Query Service should have a minimum of four cores.

Software

Java 8 and Python 2.7 in particular are required to run Cassandra and Spark. Java 11 is required to run Logi Composer. See the latest version of the *QAD Adaptive UX On-Premise Installation Guide,* available on the QAD Document Library*,* for other software prerequisites.

Analytics (PLA) License

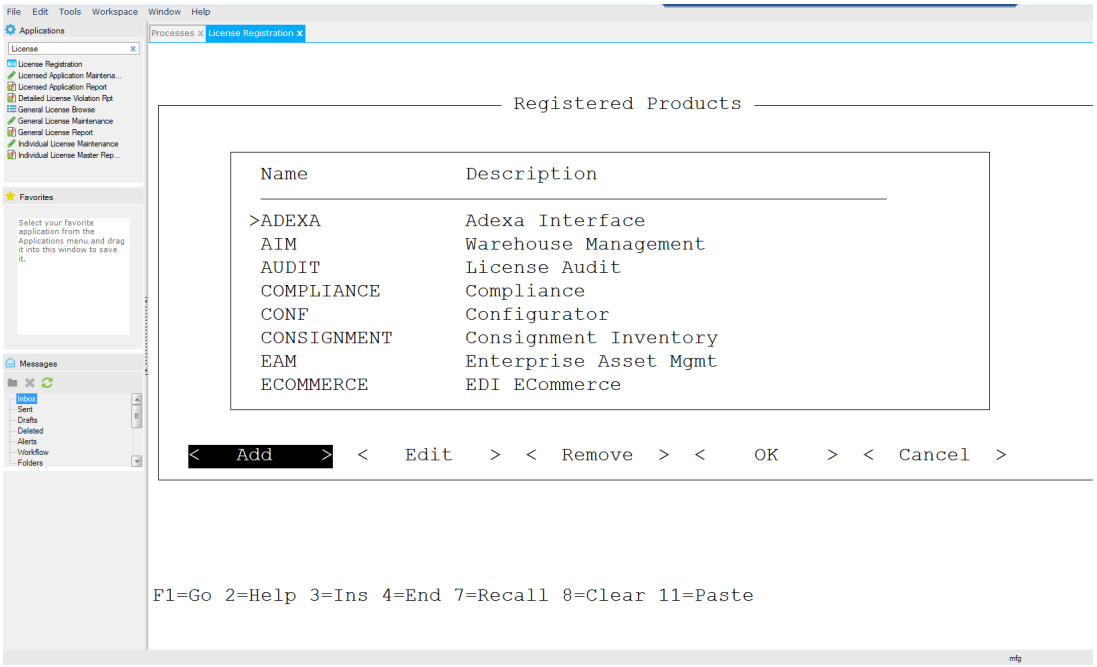
Starting with the September 2019 release, all QAD users who use Action Centers and KPIs must be registered to use the QAD Platform Analytics (PLA) license. The PLA license is provided as part of Enterprise Edition and administered in the same way as other QAD Enterprise Edition licenses. Users can be registered to use the license only with the License Registration screen in the QAD .NET UI.Licenses cannot be queried or updated in the Web UI. To avoid errors during the QAD Adaptive ERP installation, the PLA license should be present in the target environment and registered to appropriate users before installing or upgrading Adaptive ERP using YAB.

If the batch user is not registered for the PLA license using this procedure, the YAB installation/update will fail and corrective action will be needed as described later in this section.

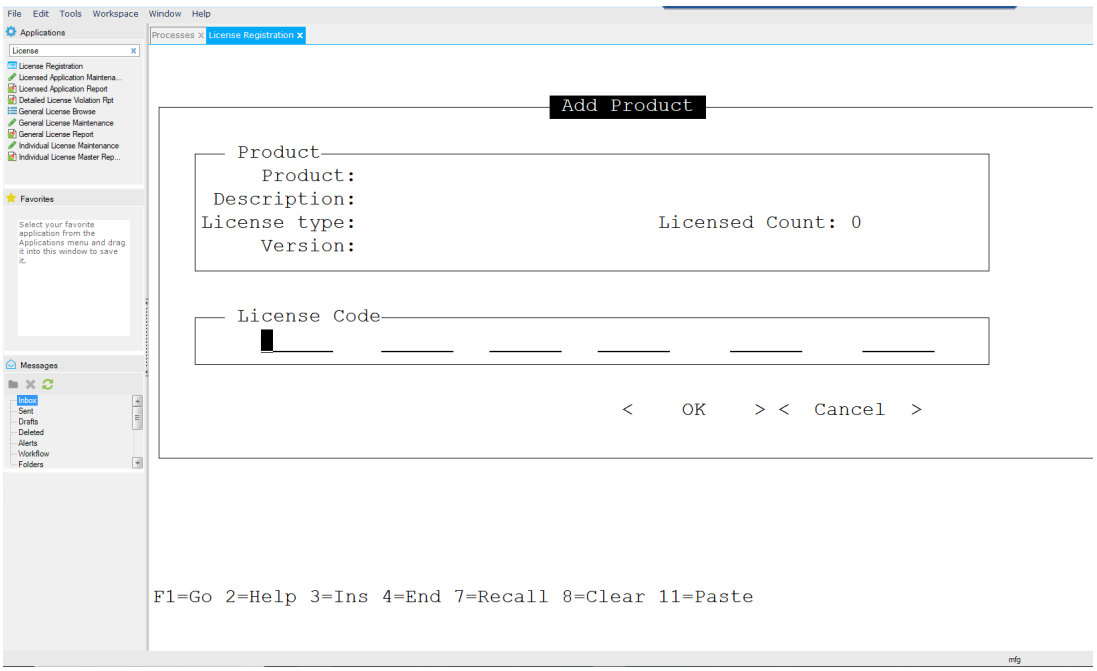
Adding PLA license and license codes

When installing or upgrading Adaptive ERP on Enterprise Edition installations prior to 2019, do the following before running the Adaptive ERP installation or upgrade.

- Log into the .NET UI as an admin user, search for the License Registration screen, and open it.
- Scroll down the list of licenses, checking for the PLA license.
- If the PLA license is already present in the list, skip to the procedure 'Registering Users for PLA License' below.
- If the PLA license is not present in the list, obtain the correct PLA license codes to use for the environment. Customer codes are obtained from QAD Global Customer Administration (GCA).
- Navigate to the Add button and press Enter.



- Enter the license codes for the PLA license using the .NET UI License Registration screen.

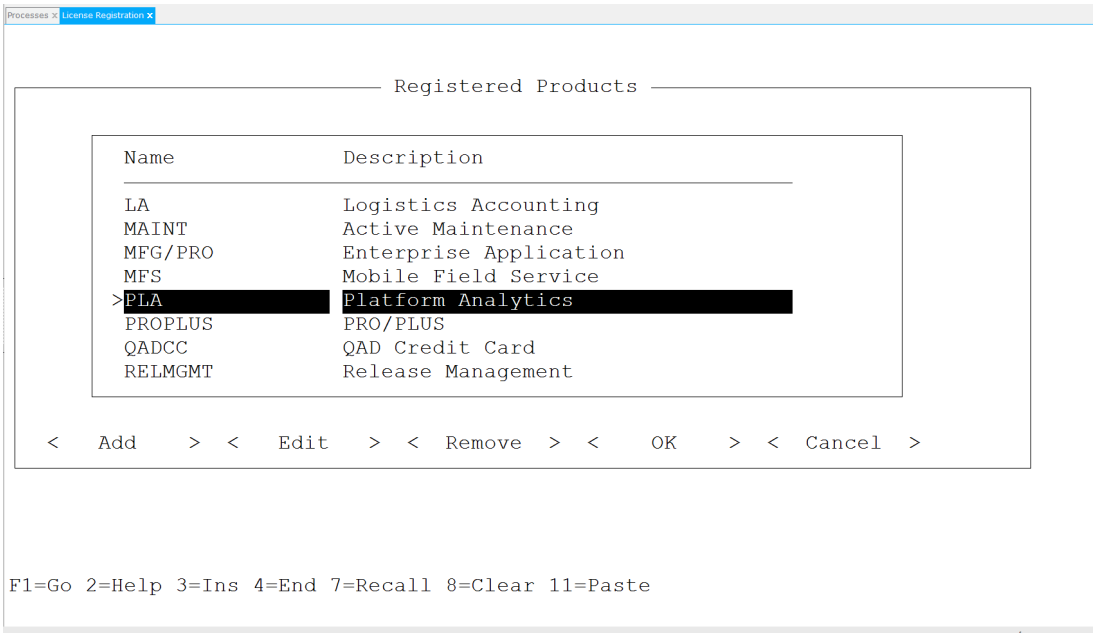


- Navigate to the OK button and press Enter.
- Confirm the changes when prompted by the UI.

Registering Users for PLA License Using NetUI

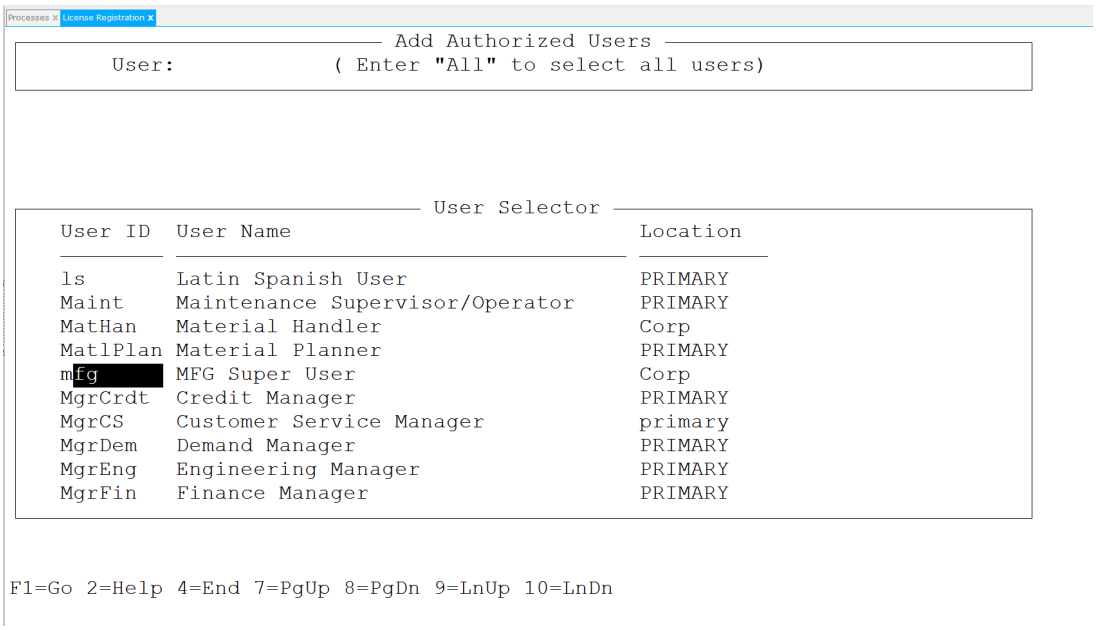
For all installations of and upgrades to the September 2019 or later release, do the following before running the Adaptive ERP installation or upgrade.

- Log into the .NET UI as an admin user, search for the License Registration screen, and open it.
- Scroll down the list of licenses, checking for the PLA license.
- If the PLA license is not present in the list, follow the previous procedure, 'Adding PLA License and License Codes.'
- Select the PLA license from the list.

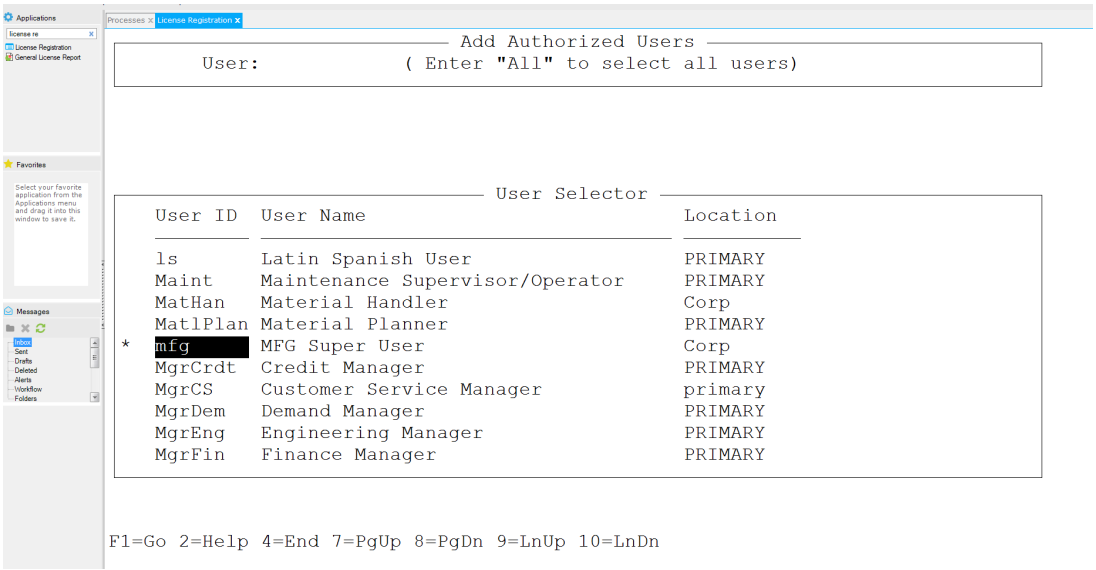


- Navigate to the OK button, and press Enter to access the list of registered users.

- To register a single user, in particular the 'batch user,' to use the PLA license, press Enter to see the list of users.



- Navigate to the user to be registered, and press Space to select the user. An asterisk will be displayed to the left of the user.

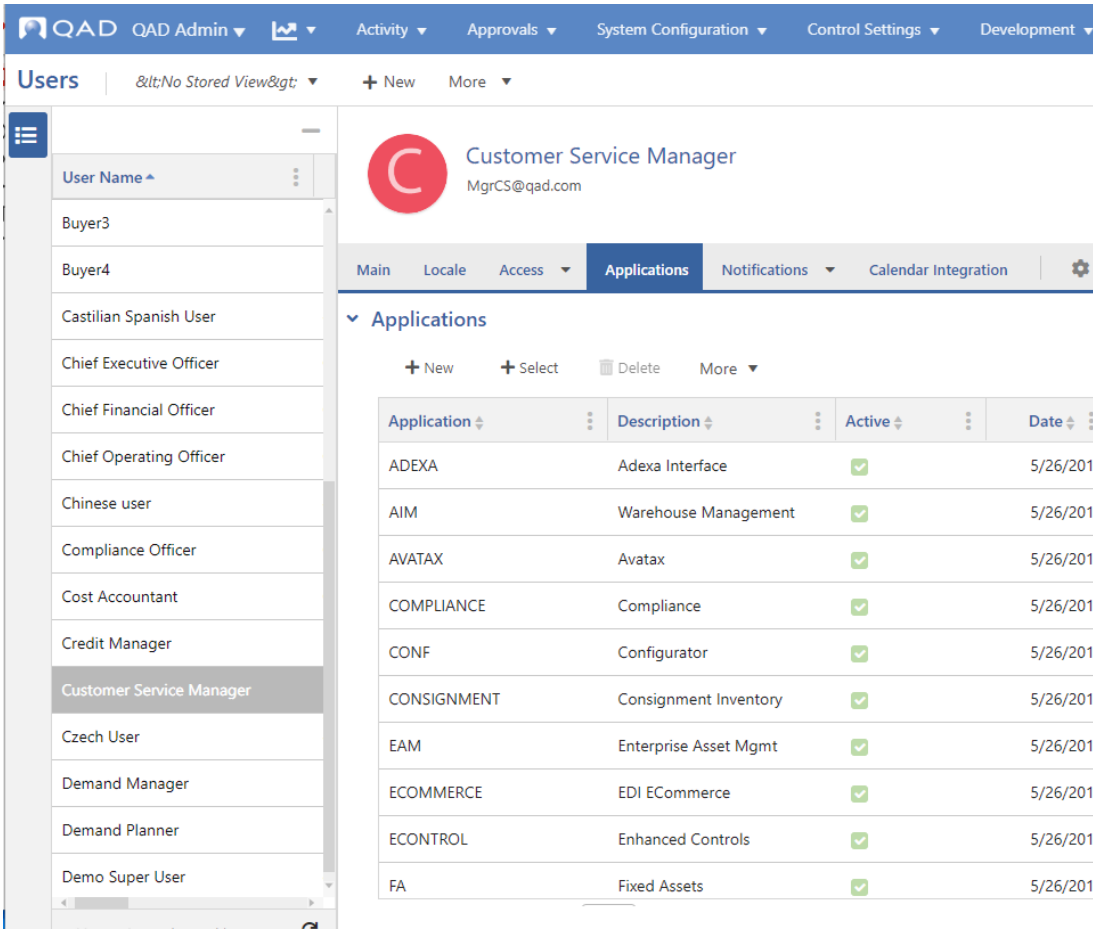


- Press the GO key to process the change, and confirm it when prompted by the UI.
- Alternatively, to register all users in the environment to use the PLA license, enter "All," press the GO key (usually F1), and confirm the action when prompted by the UI.

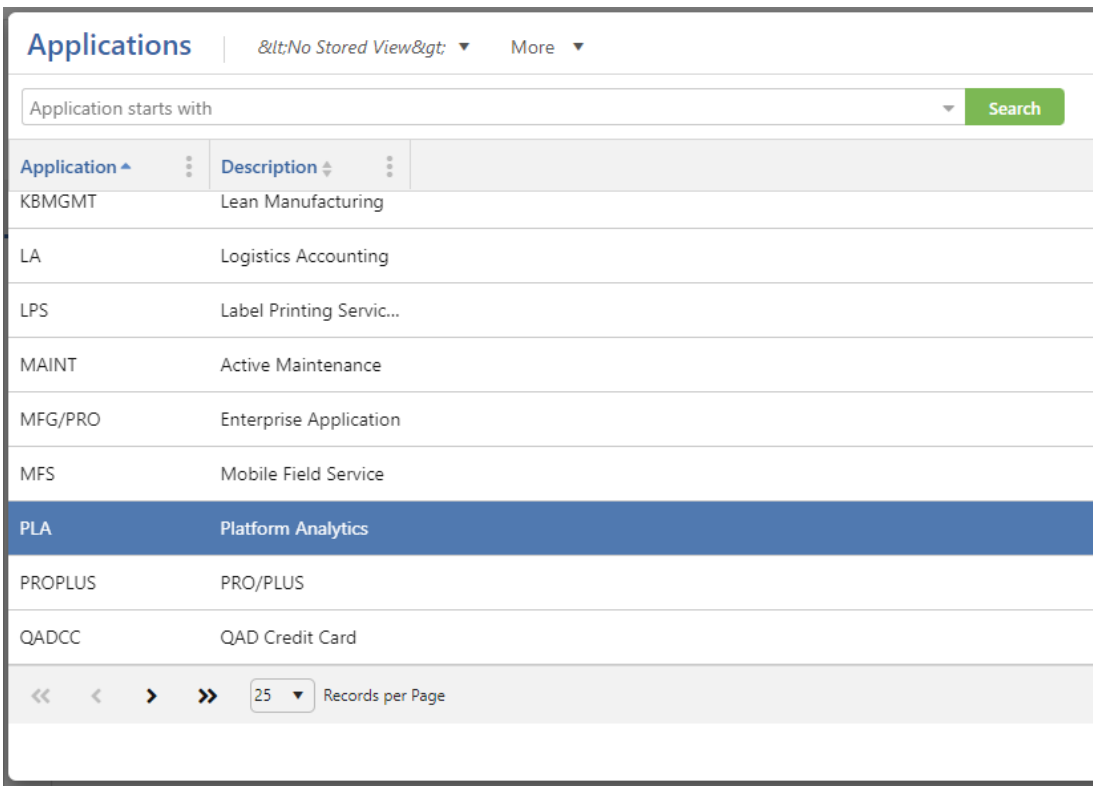
Registering Users for PLA License Using WebUI

As of the March 2022 release, individual QAD users can be registered for licenses in the WebUI as well as the NetUI. To register a user for the PLA license in the WebUI, do the following.

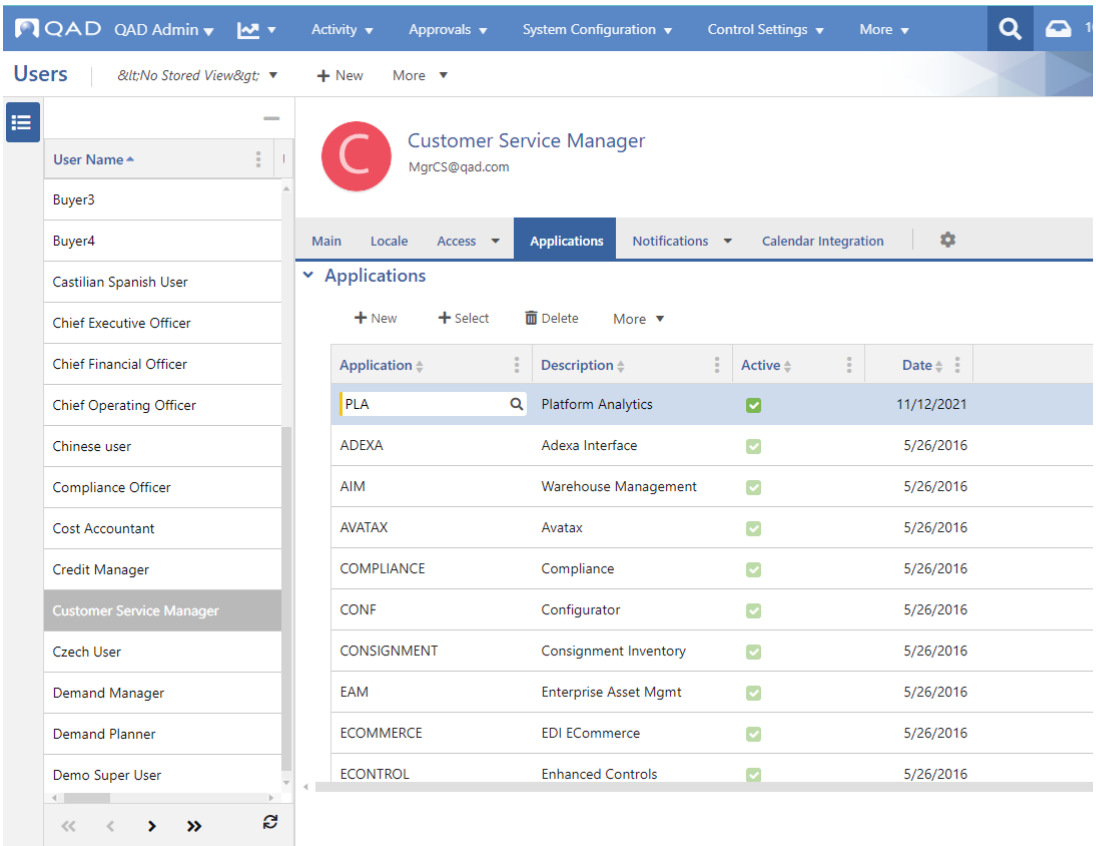
- Display the Users screen, select the user to register for the PLA license, and navigate to the Applications panel of the screen.



- Press the New button to add a new line to the Applications grid. Press the lookup icon in the Application field of the new line to display a list of applications for which the user can be registered.



- Find and select the PLA license from the list, and press OK.



- Press the Save button to register the user for the license.

Error Caused by Missing PLA License Registration for Batch User

If the preceding step to register the QAD batch user for the PLA license is omitted and Logi Platform Services is being used, the YAB installation will fail with errors in an **lps-*-referencedataview-update** step written to the YAB log as shown below.

```
DEBUG [main] HttpCallCommand - POST https://vmltdc0078.qad.com:22011/qad-central/api/analytics/lps
/deployment/artifacts?type=referenceDataview&appUri=urn:app:com.qad.assetmgmt HTTP/1.1
DEBUG [main] HttpCallCommand - HttpResponseProxy{HTTP/1.1 403 Forbidden [Server: Apache-Coyote/1.
1, Set-Cookie: JSESSIONID=6BDC450A1A4CFB07E6418836F93F494C; Path=/qad-central; Secure; HttpOnly,
Cache-Control: no-store, X-Frame-Options: SAMEORIGIN, Content-Type: application/json;charset=UTF-
8, Transfer-Encoding: chunked, Date: Wed, 14 Aug 2019 21:36:30 GMT] ResponseEntityProxy{[Content-
Type: application/json;charset=UTF-8,Chunked: true]}}
DEBUG [main] APPLY - lps-assetmgmt-referencedataview-update ERROR
ERROR [main] Main - BUILD FAILED (1:12 h)
java.lang.RuntimeException: HTTP/1.1 403 Forbidden
        at com.qad.build.java.tasks.http.HttpCallCommand.invokeToResponse(HttpCallCommand.java:
181)
        at com.qad.build.java.tasks.http.HttpCallCommand.invoke(HttpCallCommand.java:47)
        at com.qad.yab.qra.QraWebuiApiClient.submit(QraWebuiApiClient.java:115)
        at com.qad.yab.qra.QraLpsArtifactUpdateProcess.execute(QraLpsArtifactUpdateProcess.java:
171)
        ...
```

The previous example references the 'assetmgmt' app, but the specific app raising the error in other environments could be different.

This error is triggered by missing permissions for the batch user, caused by the fact that the user has not been registered for the PLA license. To correct the problem, do the following.

- Complete the steps in the section 'Adding PLA license and license codes,' if the PLA license and codes do not already exist in the environment.
- Register the batch user for the PLA license as described in the section 'Registering Users for PLA License in NetUI' or 'Registering Users for PLA License in WebUI.'
- Run the following YAB command to re-execute the failed YAB command(s) that failed in the first attempt.

```
yab -clean lps-artifact-update
```

- Re-run the YAB installation or update.

Enabling Logi Composer vs Logi Platform Services

Starting with the September 2022 release, Logi Composer is the default and preferred Logi framework used to support Action Centers. Logi Platform Services is deprecated as of September 2022 and will be retired in a future AUX release. However, it is still possible for existing QAD customers upgrading from an earlier AUX release to continue using Logi Platform Services, in order to postpone the migration of existing Action Centers to Logi Composer.

To install a September 2022 AUX environment enabled to use Logi Platform Services rather than Logi Composer, set the following YAB property before running the installation.

```
qad-analytics-core.composer.enabled=false
```

When Logi Composer is enabled, existing Action Centers not provided by QAD that were created with Logi Platform Services are automatically migrated to Logi Composer during the installation. This migration process and the required manual steps are covered in a separate section of the current guide: Logi Composer Migration Guide. This guide also contains more information about how to defer the migration to Logi Composer during the initial upgrade to AUX September 2022, and complete it sometime later.

Logi Licenses

To use Action Centers, a license is required from Logi Analytics, in addition to the PLA license required by QAD. Regardless of the Logi product being used or the AUX version, as of Aug 2022 a current Logi license must be installed in production and non-production environments, or else Action Centers and visuals cannot be used. This license is an OEM license that is automatically installed with the following AUX releases.

- Sep 2022: All releases
- Mar 2022: All releases
- Sep 2021: Patch Bundle 2 (released 27 Jan 2022) and above
- Mar 2021: Patch Bundle 3 (released 25 Apr 2022) and above, part of QAD Enterprise Platform 3.17.6 (released 18 Nov 2021)
- Sep 2020: Patch Bundle 4 (released 10 Dec 2021) and above
- Mar 2020: Patch Bundle 6 (released 14 Jan 2022) and above

For AUX environments using any of these releases, no special action is required to obtain the correct Logi license.

For environments using older AUX releases, a separate Logi License Replacement Utility must be run to update the license. For Cloud customer, this utility is run by QAD Cloud personnel. On-premise customers can download the utility with instructions from this link in the QAD Store: https://store.qad.com/content/logi-license-replacement-utility-0.

However, after installing the new license into older environments, the following steps must be excluded from the YAB update process to ensure that the new license is not automatically reverted during a subsequent YAB update.

- logi-platform-services-default-license-unassign
- logi-platform-services-default-license-assign
- logi-platform-services-default-license-import

This can done by adding the above lines to the <AUX home>/build/config/etc/process-ignore file.

Logi Platform Services Upgrades in the September 2019 Release

Starting with the September 2022 release, Logi Composer is the only option available to install in new AUX environments and the default option for existing AUX environments. While Logi Platform Services can still be used in older AUX environments that have been upgraded to September 2022, it is deprecated and will be retired in a future AUX release.

When upgrading an existing QAD Adaptive UX September 2019 environment to a service pack that includes a newer version of Logi Platform Services, several additional steps are required before running the full YAB update.

This section applies only to existing September 2019 installations with Logi Platform Services that are upgrading to a newer Logi Platform Services service pack. It does not apply to installations that did not previously contain Logi Platform Services, such as new installations or upgrades of environments that are older than September 2019. It also does not apply to QAD Adaptive UX installations of the March 2020 release or later.

Stop the Environment

First, stop the Adaptive ERP environment. Afterward, make sure that Logi Platform Services is no longer running.

```
yab stop
yab logi-platform-services-status
```

Back Up Logi Platform Services Database

If the environment contains Action Centers and visuals that have not been exported and released as part of an app, manually copy the Logi Platform Services database to a temporary file system location so its contents will not be lost during the upgrade. The Logi Platform Services database is stored in the location **<Adaptive ERP root>/servers/logi-platform-services/default/platform/db/**.

```
mkdir /mydirectory/temp
cp <Adaptive ERP root>/servers/logi-platform-services/default/platform/db/LogiDB.mv.db
/mydirectory/temp/
```

Upgrade to the new Service Pack

Run the following YAB command, which should take only a moment to complete.

```
yab logi-platform-services-default-rebuild
```

Restore Logi Platform Services Database

After the rebuild has completed, manually restore the Logi Platform Services database backup to its original location, if a backup was taken.

```
cp /mydirectory/temp/LogiDB.mv.db <Adaptive ERP root>/servers/logi-platform-services/default
/platform/db/
```

Run YAB Update

After the previous steps are complete, run the normal YAB update.

```
yab update
```

HTTPS Certificates for Logi Platform Services

This section does not apply to AUX installations using Logi Composer, only Logi Platform Services. For Logi Composer, no special procedure is required to create and deploy HTTPS certificates.

This procedure assumes that Logi Platform Services will always be accessed using HTTPS, which is secure HTTP. If the Web UI is set up to be accessed through HTTPS, then Logi Platform Services must be configured to use HTTPS also, to avoid network security errors raised at run time. By default, Logi Platform Services is installed in QAD Adaptive ERP to be accessed through HTTPS, not basic HTTP.

Creating HTTPS Certificates

To use HTTPS with Logi Platform Services, a public certificate file and a private key file are required. However, the certificate and key files in the Java Keystore (JKS) format used by Tomcat do not work with Logi Platform Services. You must either generate new files in the PEM format, a common standard for storing cryptographic data, or convert existing JKS certificate and key files to that format.

Certificate and key files are not included in the Adaptive ERP installation and are not generated by YAB, but must be created on-site specifically for each company that is installing the software. They can be created in different ways, and there is no prescribed procedure. However, below is a simple procedure that has been used to create self-signed certificates for Logi Platform Services with the help of the open source tool openssl. This procedure converts an existing JKS keystore file into certificate and key files that can be used by Logi Platform Services. In the example, the environment variable JAVA_HOME must be set to the location of a JDK 8 installation on the local server.

```
$JAVA_HOME/bin/keytool -importkeystore -srckeystore myKeystore.jks -destkeystore myKeystore.p12 -
deststoretype PKCS12  # Convert JKS to PKCS 12 syntax
openssl pkcs12 -in myKeystore.p12 -nokeys -out myKeystore.crt  # Extract certificate file in PEM
format
openssl pkcs12 -in myKeystore.p12 -nocerts -nodes -out myKeystore.key  # Extract key file in PEM
format
```

The previous steps are sufficient to create a self-signed certificate. However, to generate a certificate signed by a Certificate Authority (CA), any required intermediate certificates must also be downloaded and appended to the end of the *.crt file generated above. Below is a sample procedure to do this, once the intermediate certificate ('myInt.crt' in the example) has been obtained.

```
openssl x509 -inform der -in myInt.crt -out myInt.pem  # Convert the intermediate certificate
from binary (DER) to textual X.509 (PEM) format
cat myInt.pem >> myKeystore.crt  # Append the intermediate certificate to the certificate file
```

The root certificate for the Certificate Authority is not required, as it should already be a well-known, trusted CA.

Configuring Certificate Files and Location

Once the certificate files are available, they must be placed in a fixed location on the file system where they can be read by YAB and copied into the Logi Platform Services installation. Various YAB properties must be set describing the files so that Logi Platform Services can consume them. The required properties are described in the following table.

Property Name	Description	Default Value
logi-platform-services.default.sourcekeyfile	Full path of the filename containing the private key to be used by LogiPS.	N/A
logi-platform-services.default.sourcecertfile	Full path of the filename containing the public certificate to be used by LogiPS.	N/A
logi-platform-services.default.selfsigned	Indicates if the certificate is self-signed.	false

As with other YAB properties, these should be set in configuration.properties.

Apache Reverse Proxy Configuration

Many Adaptive ERP installations, including those hosted in the QAD Cloud, deploy an Apache web server as a reverse proxy in front of the Tomcat container to intercept and forward all web requests through a single port exposed to the internet. This kind of reverse proxy can enhance network security generally, which is not a topic covered in this document. However, in environments where an Apache reverse proxy is being used, particular configuration changes are required to support both Logi Composer and Logi Platform Services correctly.

Logi Composer

Composer uses the WebSockets protocol to support client-server communications between the web browser and its services and data connectors. Several Apache configuration changes are required to support WebSockets.

First, the Apache proxy_wstunnel_module should be enabled. To do this, uncomment the following line in <Apache home>/conf/httpd.conf.

```
LoadModule proxy_wstunnel_module modules/mod_proxy_wstunnel.so
```

In Adaptive ERP installations, the Apache configuration file <Apache home>/conf.d/vhosts.conf contains a <VirtualHost *:443> element where the HTTPS configuration settings for the virtual host are defined.

```
<VirtualHost *:443>
...
</VirtualHost>
```

Add the following rewrite settings to this element.

```
<VirtualHost *:443>
...
    RewriteEngine on
    RewriteCond %{HTTP:Upgrade} websocket [NC]
    RewriteCond %{HTTP:Connection} upgrade [NC]
    RewriteRule ^/<reverse proxy path>/?(.*) "wss://<tomcat-webui hostname>:<tomcat-webui port>
/<webapp context>/$1" [P,L]
...
</VirtualHost>
```

Below is a sample from a reverse proxy with the name "clouderp".

```
<VirtualHost *:443>
...
    RewriteEngine on
    RewriteCond %{HTTP:Upgrade} websocket [NC]
    RewriteCond %{HTTP:Connection} upgrade [NC]
    RewriteRule ^/clouderp/?(.*) "wss://vmlfwy0000.qad.com:22011/qad-central/$1" [P,L]
...
</VirtualHost>
```

Once this change is made, restart the Apache server. This is done outside of YAB, with no need to restart Tomcat or any other component of Adaptive ERP.

Logi Platform Services

In Adaptive ERP installations, an Apache configuration file defining the reverse proxy in <Apache home>/conf.d/proxy/ contains a Location element that references the URL path of the reverse proxy, in this case 'clouderp.' Below is a sample from the file conf.d/proxy/clouderp.conf in one such environment.

```
<Location /clouderp>
   ProxyPass https://vmlqad0000.qad.com:22011/qad-central
   ProxyPassReverse https://vmlqad0000.qad.com:22011/qad-central
   Header edit Set-Cookie "^(.*)/qad-central(.*)$" $1/clouderp$2
   Header edit Location "/qad-central/" "https://vmlqad0000.qad.com/clouderp/"
   AddOutputFilterByType SUBSTITUTE text/html image/svg+xml
   Substitute "s|/qad-central|/clouderp|iq"
   Substitute "s|https://vmlqad0000.qad.com:22011|https://vmlqad0000.qad.com|iq"
</Location>
```

In order to support Logi Platform Services, change the AddOutputFilterByType element in the previous example to the following, adding the MIME type 'application/com.qad.webshell.proxy+json' to the end.

```
AddOutputFilterByType SUBSTITUTE text/html image/svg+xml application/com.qad.webshell.proxy+json
```

Once this change is made, restart the Apache server. This is done outside of YAB, with no need to restart Tomcat or any other component of Adaptive ERP.

Installing Action Centers

Many QAD application packages include pre-defined Action Centers and KPIs. These standard Action Centers can be used as samples and starting points for creating custom Action Centers to suit the needs of different parts of the organization. As of the September 2022 release, the following application packages include Action Centers built using Logi Composer.

- Assetmgmt
- Custrelmgmt

- Financials
- Fixedassets
- Inventory
- Planning
- Purchasing
- Pushproduction
- Quality
- Sales
- Service
- TAM (Trade Activity Management)

In earlier releases, the QAD-provided Action Centers were built using Logi Platform Services or Logi Info.

Action Center Installation Commands

In terms of technical deployment, Action Centers and their related data fall into two parts, both of which are included in Adaptive UX apps that provide Action Centers.

- Metadata loaded into the QAD database
  - KPI metadata and related child data
  - Action Center resource records
- Metadata loaded into the Logi database
  - JSON objects loaded into Logi Composer (for Action Centers created starting with the September 2022 release)
  - JSON objects loaded into Logi Platform Services (for Action Centers created from the September 2019 through March 2022 releases)
  - XML data files copied into Logi Info (for Action Centers created before the September 2019 release)

The Action Center metadata for the QAD database is loaded for each app as part of the YAB *metadata-update* command and its sub-commands. To install the metadata for a single package, including its Action Centers, run the YAB command with the syntax *metadata-<package>-update*. For example, the Sales Action Centers are installed by the YAB command *metadata-sales-update*.

The Action Center objects for Logi Composer are loaded by the same *metadata-update* and *metadata-<package>-update* commands as the QAD data. However, they are loaded as Configuration Data rather than app-specific artifacts, as in earlier AUX versions. Once installed, they are no longer directly associated with their source apps, but are owned entirely by the users of the environment.

The Action Center objects for Logi Platform Services are loaded for each app by a set of YAB commands, each of which loads a different kind of Logi object.

- lps-<package>-tag-update
- lps-<package>-referencedataview-update
- lps-<package>-enrichmentdataview-update
- lps-<package>-visualization-update
- lps-<package>-crosstabtable-update
- lps-<package>-dashboard-update

For each app, these commands must be run in the order shown. The command *lps-artifact-update* loads all of them in the prescribed order for all apps in the environment.

The Action Center files for Logi Info are copied for all apps at the same time for each file type (dashboard, KPI, gallery) by the following YAB commands.

- action-center-dashboard-update
- action-center-kpi-update
- action-center-gallery-update

Action Centers Developed Using Logi Composer are Maintained as Configuration Data

Starting with the September 2022 release and the introduction of Logi Composer, all Action Centers and KPIs are treated as Configuration Data rather than app artifacts in most environments. In development environments only, Action Centers and KPIs are automatically associated with the app that was the active app at the time of their creation, and are exported with that app automatically. However, they are always installed as Configuration Data and can be freely maintained by AUX users regardless of their origin app. Unless overridden by a special property (see below), once installed in an environment they will not be changed or deleted by an app upgrade. This approach allows AUX users to freely adopt and tailor the Action Centers and KPIs obtained from an app without the risk that they will later be overwritten and without need to copy them into local or custom apps.

For environments that are used for app development and testing, as opposed to production, there is the need to change or delete the Action Centers and KPIs that came from previous versions of app to current versions. In order to allow new versions of these artifacts to be promoted through the steps of the app development life cycle, a property *qad-analytics-core.composer.artifacts.overwrite* has been introduced. If this property is set to true, the upgrade of an app in the AUX environment will process updates and/or deletes of Action Centers and KPIs that came from an earlier version of the same app.

Action Centers Developed Using Logi Platform Services are Installed, Updated, and Deleted

From the September 2019 release through the March 2022 release, Action Centers and KPIs are developed using Logi Platform Services only. In addition, Action Centers and KPIs that are developed in one app (for example, one of the standard QAD apps previously listed) cannot be updated or deleted by users whose active app is different. In order to use a standard QAD Action Center as the starting point for a custom version, a user can clone the Action Center and its associated KPIs into the active app using Save As and Copy functions in the Web UI. The cloned versions can then be modified as desired, eliminating the need to prevent standard QAD Action Centers from being updated. Therefore, Action Centers and KPIs beginning with the September 2019 release built using Logi Platform Services are add-updated-deleted during YAB updates in the same way as other application artifacts.

Action Centers Developed Using Logi Info are Installed But Not Updated or Deleted

In releases prior to September 2019, there is an important difference in the handling of Action Centers vs other kinds of metadata. These Action Centers, built using the older Logi Info framework, are installed by YAB, but not updated or deleted once they are installed. Standard Action Centers may be used off the shelf and modified after installation to meet local requirements. If the standard Action Centers were routinely updated as part of a system or package upgrade, local modifications would be overwritten and lost. While any application packages containing Action Centers can be upgraded, data related to existing Action Centers and KPIs is skipped during the YAB update.

To update Action Centers and KPIs inside one environment with Action Centers and KPIs of the same name that have been created in a different environment (for example, migrate updated versions of an Action Center and its associated KPIs from a development environment to production), use the KPI Migrate function in the Web UI to export the KPIs from the source environment and import them into the target environment. There is no similar function to export and import an Action Center, so a modified Action Center must be updated manually in the target environment. However, updating an Action Center using Logi Info consists mainly of removing/adding/rearranging dashboard panels, and this is usually a quick process.

When updating apps that were developed outside the organization (for example, a recent release of a previously installed QAD package), newer versions of Action Centers and KPIs that already exist in the target environment are not updated for the reasons previously mentioned. In this case, the source environment in which the Action Centers and KPIs were defined is not available and the KPI Migrate function cannot be used to export and import them. If the newer versions of these predefined Action Centers and KPIs are needed, please contact QAD Support for assistance.

When updating apps that include Action Centers and KPIs new to the target environment, no special steps are needed. The new Action Centers and KPIs are installed automatically as part of the YAB update.

Exporting Action Centers

The App Name field shows the app in which the KPI data is stored. The field is read-only, and is set to the app selected as your active app in the My Developer Settings screen. Action Centers and KPIs are automatically exported with that app using the YAB *app-export* command. The output of the export written to the specified directory includes the following files related to Action Centers. These files are generated by the Action Center infrastructure and should not be manually edited.

When you migrate a KPI from a source environment to a target environment, the migration does not include the browse that is used as the data source. You must ensure the browse used as the data source already exists in the target environment and that it has the same fields in its definitions. You can migrate browse definitions with the import/export tool in Browse Maintenance in the QAD .NET UI.

Action Centers Created In September 2022 Release (Logi Composer)

- ac/lc-artifact/analytics/ac/ sub-directory
  - <Directory for each Action Center, identified with a UUID value>
    - **composer/**: Directory of Composer metadata for the Action Center.
      - **dashboard.json**: Composer metadata for the Action Center.
    - **kpi/**: Directory of visuals and associated KPIs contained in the Action Center.
      - <Directory for each KPI used in the Action Center, identified with a UUID value>
        - **composer/**: Directory of Composer metadata for the KPI.
          - **custom-metrics.json**: Composer metadata for any custom metrics defined for the KPI.
          - **fields.json**: Composer metadata for the data fields defined for the KPI.
          - **visualization.json:** Composer metadata for all visuals defined for the KPI.
        - **KpiMetadata.json**: Database records defining the KPI. These records define the data fields, filters, and domains/entities significant for a particular KPI.
    - **PECActionCenter.json**: Database records defining the Action Center. These records allow permissions to access particular Action Centers to be granted or revoked based on role.

Action Centers Created In September 2019 Through March 2022 Releases (Logi Platform Services)

- data/analytics/ sub-directory
  - **D-*.xml** files: Database records defining an Action Center. These records allow permissions to access particular Action Centers to be granted or revoked based on role.
  - **K-*.xml** files: Database records defining a KPI. These records define the data fields, filters, and domains /entities significant for a particular KPI.
- ac/lps/ sub-directory
  - **dashboard/**: Directory of LogiPS database contents for Action Centers in JSON files.
  - **enrichmentdataview/**: Directory of LogiPS database contents for enrichment dataviews in JSON files.
  - **referencedataview/**: Directory of LogiPS database contents for reference dataviews in JSON files.
  - **tag/**: Directory of LogiPS database contents for tags in JSON files.
  - **visualization/**: Directory of LogiPS database contents for visuals in JSON files.

Action Centers Created Before September 2019 Release (Logi Info)

- data/analytics/ sub-directory
  - **D-*.xml** files: Database records defining an Action Center. These records allow permissions to access particular Action Centers to be granted or revoked based on role.
  - **K-*.xml** files: Database records defining a KPI. These records define the data fields, filters, and domains /entities significant for a particular KPI.
- ac/dashboard/ sub-directory
  - **Dashboard-*.xml** files: Logi Info file defining the layout and contents of an Action Center.
- ac/gallery/ sub-directory
  - **Gallery.xml** file: Logi Info file containing information about the visuals in the exported package published for use in Action Centers.
- ac/kpi/ sub-directory
  - **AGState-*.xml** files: Logi Info file defining the layout and content of the visuals displayed from an expanded Action Center panel, or by pressing the Visuals button for a particular KPI in the KPI screen.

YAB Commands

Logi Composer

To obtain a list and description of all YAB commands that can be used to monitor and control Logi Composer, run the following command.

```
yab help logi-composer-
```

Most of the Logi Composer commands documented in the YAB help are run only by YAB, and would not normally be run directly from the command line. However, some of them can be useful to system administrators. The following YAB commands would normally be run from the command line for routine monitoring and control purposes. Any of them can be run without the need to stop/start Tomcat or other Adaptive ERP components.

YAB Command	Description	Remarks
postgresql-default-start	Starts the 'default' instance of the PostgreSQL database used by Composer.	
postgresql-default-status	Checks the status of the 'default' instance of the PostgreSQL database used by Composer.	
postgresql-default-stop	Stops the 'default' instance of the PostgreSQL database used by Composer.	
postgresql-default-restart	Restarts the 'default' instance of the PostgreSQL database used by Composer.	
logi-composer-default-start	Starts the 'default' instance of Logi Composer.	
logi-composer-default-status	Checks the status of the 'default' instance of Logi Composer.	
logi-composer-default-stop	Stops the 'default' instance of Logi Composer.	
logi-composer-default-restart	Restarts the 'default' instance of Logi Composer.	

Logi Platform Services

To obtain a list and description of all YAB commands that can be used to monitor and control Logi Platform Services, run the following command.

```
yab help logi-platform-services-
```

To obtain a list and description of all YAB commands that are used to install and extract Action Centers and related objects to/from Logi Platform Services, run the following command. The YAB commands described are run by YAB during Action Center deployment.

```
yab help lps-
```

Most of the Logi Platform Services commands documented in the YAB help are run only by YAB, and would not normally be run directly from the command line. However, some of them can be useful to system administrators. The following YAB commands would normally be run from the command line for routine monitoring and control purposes. Any of them can be run without the need to stop/start Tomcat or other Adaptive ERP components.

YAB Command	Description	Remarks
logi-platform-services-default-application-start	Starts the application service for the Logi Platform Services 'default' instance.	The data service must be started before the application service is started.
logi-platform-services-default-application-status	Returns status of the application service for the Logi Platform Services 'default' instance.	
logi-platform-services-default-application-stop	Stops the application service for the Logi Platform Services 'default' instance.	The application service must be stopped before the data service is stopped.
logi-platform-services-default-data-start	Starts the data service for the Logi Platform Services 'default' instance.	The data service must be started before the application service is started.
logi-platform-services-default-data-status	Returns status of the data service for the Logi Platform Services 'default' instance.	
logi-platform-services-default-data-stop	Stops the data service for the Logi Platform Services 'default' instance.	The application service must be stopped before the data service is stopped.
logi-platform-services-default-start	Starts the 'default' Logi Platform Services instance.	Combines the logi-platform-services-default-data-start and logi-platform-services-default-application-start commands.
logi-platform-services-default-status	Returns the status of the 'default' Logi Platform Services instance.	
logi-platform-services-default-stop	Stops the 'default' Logi Platform Services instance.	Combines the logi-platform-services-default-application-stop and logi-platform-services-default-data-stop commands.
logi-platform-services-default-restart	Stops and immediately re-starts the 'default' Logi Platform Services instance.	Tomcat-webui does not have to be restarted when LogiPS is restarted.
logi-platform-services-license-info	List the LogiPS license(s) installed in the Logi Platform Services 'default'	Useful for checking the presence or type of a LogiPS license in case of license-related errors.

	instance.	
logi-platform-services-license-assign	Assigns or re-assigns and installs a LogiPS license to the Logi Platform Services 'default' instance.	The type of license assigned depends on whether the environment type is development, test, or production.
logi-platform-services-license-unassign	Un-assigns and deletes the currently assigned LogiPS licenses from the Logi Platform Services 'default'instance.	May be needed if the wrong kind of Logi license was installed for any reason, or if the production environment is being deleted or its server de-commissioned. In the latter case, failure to un-assign the license would trigger the purchase of an extra LogiPS production license unnecessarily.

Logi Info

To obtain a list and description of all YAB commands that can be used to deploy and extract Action Center files used by Logi Info, run the following command, which gives details for the three types of Action Center files: dashboard files, gallery, and KPI files.

```
yab help action-center-
```

Cassandra

To obtain a list and description of all YAB commands that can be used to administer Cassandra, run the following command.

```
yab help cassandra-
```

For a list of all Cassandra-related settings, including the YAB commands, run the following command.

```
yab help cassandra
```

Following are the commands that would most commonly be run from the command line.

YAB Command	Description	Remarks
cassandra-default-status	Checks the status of the 'default' Cassandra instance.	
cassandra-default-start	Starts the 'default' Cassandra instance.	
cassandra-default-stop	Stops the 'default' Cassandra instance.	
cassandra-default-restart	Stops and re-starts the 'default' Cassandra instance.	Spark and tomcat-webui do not have to be restarted when Cassandra is restarted.
cassandra-default-nodetool	Runs the Cassandra nodetool command-line utility.	Run 'yab cassandra-default-nodetool -command:help' for a list of subcommands in the nodetool utility, and 'yab cassandra-default-nodetool -command:<subcommand> to run any of them. For more detailed background about each one, see the Cassandra web pages.

Spark

To obtain a list and description of all YAB commands used to administer Spark, run the following command.

```
yab help spark-
```

For a list of all Spark-related settings, including the YAB commands, run the following command.

```
yab help spark
```

Most of the Spark commands documented in the YAB help will be rarely used, especially given that all Adaptive UX releases deploy Spark in a non-clustered manner on a single server. Following are the ones that would most commonly be run from the command line.

YAB Command	Description	Remarks
spark-status	Checks the status of all Spark nodes and processes.	
spark-start	Starts the Spark master and slave (worker) processes.	
spark-stop	Stops the Spark master and slave (worker) processes.	
spark-restart	Stops and re-starts the Spark master and slave (worker) processes.	If Spark is restarted, tomcat-webui must also be restarted in order to restore Query Service connections to Spark.

Action Center Security

This section describes various features, configuration settings, and considerations related to security in the Action Center and Query Service.

Action Centers

Roles and Permissions

Action Centers are part of the QAD Web UI and are displayed on the Web UI menu. The permissions required to access them in different ways are restricted by role using the same security infrastructure as other Web UI screens. For detailed information on setting permissions, see the *QAD Security Administration Guide,* available on the QAD Document Library.

Permissions are granted by role to create Action Centers, and to view and delete particular Action Centers. "Sharing" permissions can also be granted by role that allow authorized users to add, replace, and delete visuals in the common gallery. In addition, the user who created a particular Action Center always has full access permissions to it, regardless of his/her role. In this respect, Action Centers are different from other secured resources in Adaptive ERP. For more information on Action Center permissions, see the online help for the QAD Web UI (https://documentlibrary.qad.com/help/webui/2022_1/en-US/index.html).

Domain-Entity-Site Membership

Another aspect of Action Center security is the membership of users within particular domains, financials entities, and sites. This type of security is part of the common Web UI security infrastructure, and is not covered in this document. However, domain, entity, and site membership are used by the Action Centers to automatically filter the data that end users can view. For example, two users have permissions to view a particular Action Center, but User 1 is a member of domain 10USA and User 2 is not. In this case, both users would see the same panels and visuals in the Action Center, but the visuals shown to User 1 would include data from domain 10USA, whereas the visuals shown to User 2 would not.

Logi Composer Authentication

Logi Composer has its own security model that has been integrated with QAD Adaptive UX. It is generally similar to the approach used to integrate with Logi Platform Services (see below), but with one major difference: all QAD userids registered for an analytics (PLA) license are automatically created as Composer users, with their security groups and permissions within Composer set according to their QAD role permissions. This approach requires that Composer user information be synchronized automatically with important changes affecting the corresponding QAD user, but provides improved on-line performance for the AUX users displaying Action Centers in the WebUI.

System Users

Composer has two built-in users required to maintain the system: "admin" and "supervisor". These user identities are used internally for system configuration and to communicate with Composer, not for on-line Web UI Action Center access. Their passwords are set at installation time by YAB and can be changed after installation. Changing the default passwords are strongly encouraged in order to keep the system more secure. The YAB properties used to set their values are as follows.

```
logi-composer.default.admin.password=Password1!
logi-composer.default.supervisor.password=Password1!
```

The passwords must be over eight characters long with a mix of lowercase characters, uppercase characters, numbers, and special characters. In AUX environments configured as secured, they are encrypted by the Key Management Service (KMS).

Trusted Access Authentication

Although QAD users are maintained automatically as Composer users, they are not authenticated in Composer with individual passwords. Instead, a single sign-on mechanism is used in which each user obtains a Composer access token at the time of Web UI login, for Action Center use during the login session. This trusted access authentication mechanism is automatically applied and requires no system administration effort. It requires only a client ID and client secret code pair maintained in both Composer and the Web UI, which is generated by YAB automatically at installation time. For reference, the YAB properties in which these values are stored are as follows:

```
logi-composer.default.trusted-access-client-api.request.client_id
logi-composer.default.trusted-access-client-api.request.client_secret
```

The client secret is generated at installation time based on an encryption algorithm, and is specific to each environment. It can also be re-generated if needed by running the following YAB command, although this should not be required:

```
logi-composer-default-trusted-access-client-create
```

In AUX environments configured as secured, the client ID and client secret are encrypted by the Key Management Service (KMS). Whenever the client secret is re-generated, both tomcat-webui and Composer must be restarted.

Logi Platform Services Authentication

Starting with the September 2022 release, Logi Composer is the only option available to install in new AUX environments and the default option for existing AUX environments. While Logi Platform Services can still be used in older AUX environments that have been upgraded to September 2022, it is deprecated and will be retired in a future AUX release.

Logi Platform Services has its own security model that has been integrated with QAD Adaptive UX, with the goal of minimizing the amount of overlapping security data maintained across the two systems. It supports two authentication mechanisms used by the Action Centers functionality of QAD Adaptive UX: native user authentication and trusted access authentication.

Native User Authentication

LogiPS can maintain its own information and credentials for individual users, without the need for those users to be QAD users that are recognized in QAD Adaptive UX. In fact, in order to install LogiPS, there must be at least one "admin" user with full permissions to access and modify any data maintained in the LogiPS database. This "admin" user is used by QAD Adaptive UX to make background API calls into Logi Platform that are not requesting information on behalf of QAD on-line users—for example, by YAB at installation/update time in order to configure the LogiPS environment. However, because LogiPS has been embedded into QAD Adaptive UX all LogiPS end users must be QAD end users, with the QAD users, roles, and permissions controlling both QAD Adaptive UX and LogiPS usage. In general, Logi Platform Services and the Action Centers that it contains can be accessed only through the Web UI, not by logging into LogiPS directly. For this reason, no users except for the required "admin" user are created or maintained natively in LogiPS. Instead, the trusted access authentication mechanism described later in this document is used to enable QAD users to access LogiPS in a seamless manner, subject to the role-based permissions and domain-entity-site membership maintained in QAD Adaptive UX.

The credentials for the LogiPS "admin" user are set at installation time by YAB and can be changed after installation. Changing the default password is strongly encouraged in order to keep the system more secure. The username "admin" should not be changed. The YAB properties with default values are as follows:

```
logi-platform-services.default.username=admin
logi-platform-services.default.password=password
```

The password can be changed using the following YAB command:

```
yab logi-platform-services-default-password-update
```

Whenever the password is changed, both tomcat-webui and Logi Platform Services must be restarted.

Trusted Access Authentication

QAD user IDs are not maintained separately in LogiPS. Instead a single sign-on mechanism is used, where the Web UI passes required user information to LogiPS and authenticates a QAD 'trusted user' on the fly whenever needed, passing a client secret to LogiPS specific to the installation in order to vouch for the user's authenticity. The Web UI also passes important information about the user's permissions to LogiPS, so that the appropriate security rules can be defined and enforced in LogiPS to prevent users from retrieving unauthorized business data from Adaptive Applications.

This trusted access authentication mechanism is automatically applied and requires no system administration effort. It requires only a client ID and client secret code pair maintained in both LogiPS and the Web UI, which is generated by YAB automatically at installation time. For reference, the YAB properties in which these values are stored are as follows:

```
logi-platform-services.default.clientid=analytics
logi-platform-services.default.clientsecret=eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.
Fr4TBVdbNNNBnwdB7IAPUuXoVO5K-pT7KDL8SlAFjtMolXShXjg6cg.Kr5vg5dre2iKkMjlZByZEg.
KO3UzPUve149bmPTglKSCn-yahkEEIq7x-xgL-
3JtmeFs2B3do5mnXg6SrZgB59f_wG_Gk010YD4elkgzHbwLbekW0glns09MRwyDoYo2Ck.L04Buq0IrsPOSenuPA-UCA
```

The client secret is generated at installation time based on an encryption algorithm, and is specific to each environment. It can also be re-generated if needed by running the following YAB command, although this should not be required.

```
logi-platform-services-default-clientsecret-update
```

Whenever the client secret is re-generated, both tomcat-webui and Logi Platform Services must be restarted.

Logi Info Authentication

> As of the September 2021 release, Logi Info is not supported and KPIs created using Logi Info can no longer be displayed.

Logi Info is deployed as a separate Tomcat web application, named 'qad-dashboards' by default, but is intended to be accessed only from within the Web UI. Users are not allowed to access Logi Info or display Action Centers without first logging into the Web UI. To ensure that all access to the Action Centers is restricted to Web UI sessions, Logi *SecureKey authentication* is enabled.

With SecureKey authentication enabled, every Web UI request to display Action Centers or other Logi Info views is preceded by a server-to-server HTTP 'handshake' call from the Web UI webapp to the Logi Info webapp requesting a valid SecureKey token. A token is then returned to the Web UI, allowing the Web UI to include Action Center displays through a subsequent request. This SecureKey authentication handshake is processed quickly and is invisible to end users. While it is enabled by default and should require no manual installation or configuration steps, the properties controlling the processing are summarized here.

SecureKey authentication is enabled in the file `qad-analytics-core.properties` by the following property:

```
qad-analytics-core.logiSecureKeyEnabled=true
```

It is also enabled in the Logi Info configuration file `_Definitions/_Settings.lgx` in the XML Security element.

```
<Security AuthenticationClientAddresses="0.0.0.0 255.255.255.255" AuthenticationSource="
SecureKey" RestartSession="True" SecurityEnabled="True"/>
```

These properties should not be changed.

Query Service Security

Spark ThriftServer Authentication

Starting in the September 2019 release, a new component of Spark called the ThriftServer has been introduced into the Query Service. The ThriftServer allows Logi Platform Services to call the Query Service directly to retrieve required KPI and browse data using SQL requests. The ThriftServer exposes a network port that can be accessed by Logi Platform Services through a JDBC connection. The port used by the ThriftServer is assigned by YAB at installation time, and stored in the following property.

```
spark-thriftserver.default.port
```

Because the ThriftServer port allows browse results from Adaptive Applications to be retrieved using remote SQL clients with a valid JDBC connection, it must be secured to prevent unauthorized access. Because only Logi Platform Services needs to connect to this port, QAD recommends that outside access to it be blocked by a network firewall. In addition, access to the port is secured by username-password credentials that can be set in the following YAB properties:

```
spark-thriftserver.default.username
spark-thriftserver.default.password
```

YAB assigns default values to these properties. Changing the default password is strongly encouraged in order to keep the system more secure. The password can be changed in a YAB update, following by a restart of tomcat-webui and Logi Platform Services.

Cassandra Authentication

By default, authentication to access the Cassandra data lake is disabled. Whether authentication is enabled or disabled has no direct effect on Action Center users. However, the lack of authentication exposes security holes regarding access to the browse data that is stored in Cassandra. In particular, a user with the ability to run the Cassandra shell *cqlsh*, described in the Action Center Maintenance and Troubleshooting section, could connect to the data lake and view/update /delete the browse data without restriction using SQL commands. For this reason, it is strongly recommended to enable Cassandra authentication using YAB, so that valid username-password credentials are required to access any of its data.

To enable Cassandra authentication, set the following YAB property:

```
cassandra.default.node.main.authenticator=PasswordAuthenticator
```

Once Cassandra authentication is enabled, the Cassandra user and password are set to "qad" and "qad" respectively by default. Change these to more secure values for the installation by setting the following YAB properties.

```
cassandra.default.user
cassandra.default.password
```

Run a YAB update to process these changes.

Spark Web UI Access

Spark provides a native web user interface, completely separate from Action Centers, that can be used by system administrators to monitor the jobs and tasks launched and executed internally by Spark. This UI is described briefly in the Action Center Maintenance and Troubleshooting section of this document. It is a single set of integrated web pages, but the pages are actually made up of three UIs that can be accessed through separate network ports. While potentially useful, several of these UIs expose security risks to the run-time environment. The risks and the means of reducing or eliminating them through configuration are described in the following sections.

Driver UI

The 'driver' is the Query Service itself running inside Tomcat, which creates a Spark application by communicating to the stand-alone Spark cluster manager, called the 'master.' This UI allows users to view all Spark environment settings, both those set by YAB and those internal to Spark, which include internal Spark passwords. It also displays 'kill' hyperlinks that allow users to stop in-process Spark jobs, which disrupt Action Center processing. If a Spark job is killed in this manner, the Tomcat instance hosting the Web UI has to be restarted to ensure that the Action Centers work properly.

The port used to access the driver UI is configured by the following YAB property:

```
qad-qraview.spark.ui.port
```

The driver UI is enabled by default. To disable it entirely, set the following YAB property:

```
qad-qraview.spark.ui.enabled=false
```

The ability to kill Spark jobs from the driver UI is controlled by the following YAB property, set to `false` by default:

```
qad-qraview.spark.ui.killEnabled=false
```

Master UI

The "master" runs in a separate Spark JVM process with responsibility for dispatching and tracking the work across a cluster of workers, which in the current release is only a single worker node. Like the driver UI, it displays 'kill' hyperlinks that allow users to stop in-process Spark jobs, which disrupt Action Center processing.

The port used to access the master UI is configured by the following YAB property:

```
spark.masterdefault.env.webui.port
```

The master UI is automatically enabled and cannot be disabled. However, the ability to kill Spark jobs from the master UI is controlled by the following YAB property, set to `false` by default:

```
spark.masterdefault.properties.spark.ui.killEnabled=false
```

To prevent all use of the master UI, the network firewall must be configured to block access to its port.

Worker UI

The "worker" runs in a separate spark JVM process and performs queries-processing based on requests from the master node. Unlike the other Spark UIs, the worker UI does not expose any sensitive data or affect Spark processing.

The port used to access the worker UI is configured by the following YAB property. It is enabled by default and cannot be disabled.

```
spark.slavedefault.env.webui.port
```

To prevent all use of the worker UI, the network firewall must be configured to block access to its port.

Logi Database Security

Logi Composer Database

Logi Composer has two separate PostgreSQL databases, zoomdata and zoomdata-qe, to store all its metadata. These databases contain internal representations of the Action Centers, visuals, and KPIs, in addition to system configuration data. Composer connects to the databases through a single configurable port secured with login credentials.

The PostgreSQL databases are not accessed outside of Logi Composer. The port and connection password are maintained using YAB in the following properties.

```
postgresql.default.port
postgresql.default.roles.admin.password
```

The username credential to connect to PostgreSQL is always "admin".

Logi Platform Services Product Database

Logi Platform Services includes a separate "product database" (PDB) to store all its metadata. This database contains internal representations of the Action Centers, visuals, and KPIs, in addition to system configuration data. It is relational, implemented using the H2 database engine (see https://www.h2database.com/). The Logi Data Service within LogiPS connects to the PDB through a single configurable port, secured with login credentials.

The PDB is not accessed outside of LogiPS. Neither YAB nor QAD Adaptive UX ever reads or writes its contents directly, and there would be no reason for its connection port to be exposed outside of the enterprise firewall. While this port can be configured by YAB, the database connection credentials are native to LogiPS and not maintained using YAB.

The port used by the PDB for client connections is stored in the following YAB property, dynamically assigned by YAB at installation time.

```
logi-platform-services.default.service.data.h2.port
```

The PDB connection credentials are automatically set by LogiPS at installation time and would normally never need to be changed. However, LogiPS provides a command line utility `dbPassword.sh` that can be used to change them, if needed. The help text for using this utility is shown below:

```
dbPassword: Change platform MQ or DB password

Usage

-v, --verbose <number>    Provide extensive output (0-3)
--help                    Print usage instructions
-o, --offline             Invoke command in offline mode
-c, --MQ                  Change Message Queue password
-d, --DB                  Change Database password
```

The `-d` option above is used to change the PDB password. Logi Platform Services should be stopped before using the utility. After the utility is run, Logi Platform Services must be restarted.

Logi Network Security

Logi Composer Network Ports

Starting with the September 2022 release and the introduction of Logi Composer into the QAD Adaptive UX technology stack, various network ports required by Composer and automatically configured by YAB. These ports have different purposes within the Logi architecture with different security considerations. They are summarized in the following table.

Purpose of Port	YAB Property	Security Approach	Remarks	Required Access to Port
HTTP or HTTPS client access to Composer UI (URL)	`logi-composer.default.url`	Configurable username-password credentials for 'admin' and 'supervisor' users, managed by YAB.	Native Composer UI. Not used in AUX. Should be accessed only when required for troubleshooting purposes.	Sysadmin /Cloud personnel only.
Zoomdata microservice discovery	`logi-composer.default.microservice.zoomdata.configuration.discovery.registry.port`	No configured credentials, internal only. Should be secured at firewall level.	Internal to Logi Composer. The same port is used for all microservices.	Internal to Logi Composer.
Zoomdata query engine microservice discovery	`logi-composer.default.microservice.zoomdata-query-engine.configuration.discovery.registry.port`	Same as above.	Same as above.	Same as above.
SparkSQL microservice discovery	`logi-composer.default.microservice.zoomdata-edc-sparksql.configuration.discovery.registry.port`	Same as above.	Same as above.	Same as above.
PostgreSQL microservice discovery	`logi-composer.default.microservice.zoomdata-edc-postgresql.configuration.discovery.registry.port`	Same as above.	Same as above.	Same as above.
PostgreSQL zoomdata database access (URL)	`logi-composer.default.microservice.zoomdata.configuration.spring.datasource.url`	Internal username-password credentials, automatically configured by YAB	Internal to Logi Composer. The same port is used for both PostgreSQL databases. See earlier section describing PostgreSQL for more details.	Internal to Logi Composer.
PostgreSQL zoomdata-qe database access (URL)	`logi-composer.default.microservice.zoomdata-query-engine.configuration.spring.qe.datasource.jdbcUrl`	Same as above.	Same as above.	Same as above.

As shown in this table, all the ports are internal to Logi Composer except for the HTTP/HTTPS port for the Composer UI,

Logi Platform Services Network Ports

In the versions of QAD Adaptive UX where Logi Platform Services is used, various network ports required by LogiPS are automatically configured by YAB. These ports have different purposes within the Logi architecture with different security considerations. They are summarized in the following table.

Purpose of Port	YAB Property	Security Approach	Remarks	Required Access to Port
HTTPS client access to LogiPS	`logi-platform-services.default.service.application.webserver.sslport`	Native Logi Platform ('native user') authentication and Web UI ('trusted user') authentication	Used for Logi Platform API access	tomcat-webui server of QAD Adaptive UX only
HTTP client access to LogiPS	`logi-platform-services.default.service.application.webserver.port`	Native Logi Platform ('native user') authentication and Web UI ('trusted user') authentication	Used for Logi Platform API access, but disabled by default in favor of HTTPS	tomcat-webui server of QAD Adaptive UX only
H2 (PDB) database access	`logi-platform-services.default.service.data.h2.port`	Internal username-password credentials, automatically configured by LogiPS and not managed by YAB	Internal to Logi Platform. See earlier section describing the PDB for more details.	Internal to LogiPS
Embedded LDAP access	`logi-platform-services.default.service.data.ldap.port`	Configurable username-password credentials, with password encrypted in file by LogiPS and not managed by YAB	Internal to Logi Platform, little need to change password	Internal to LogiPS
ActiveMQ access - Stomp	`logi-platform-services.default.service.data.stomp.`	Internal username-password credentials, automatically configured by LogiPS and not managed by YAB	Internal to Logi Platform, little need to change password	Internal to LogiPS

	`port`			
ActiveMQ access - Openwire	`logi-platform-services.default.service.data.openwire.port`	No authentication is needed, as protocol is binary and communications are internal to ActiveMQ transport within LogiPS	Internal to Logi Platform, little need to change password	Internal to LogiPS

As shown in this table, all the ports are internal to Logi Platform Services except for the HTTPS and HTTP service ports, which are accessed from the tomcat-webui server. None of them require direct access by client browsers.

Action Center Configuration

The Configuration section describes important details about the configuration of Action Centers and the Query Service components that support them. Most configuration properties are maintained using YAB. General reference information on any of the properties can be found by running the command:

```
yab help <property name>
```

This guide includes details about only a portion of the properties used to configure Action Centers. For descriptions of most of them, run the command:

```
yab help qad-analytics-core
```

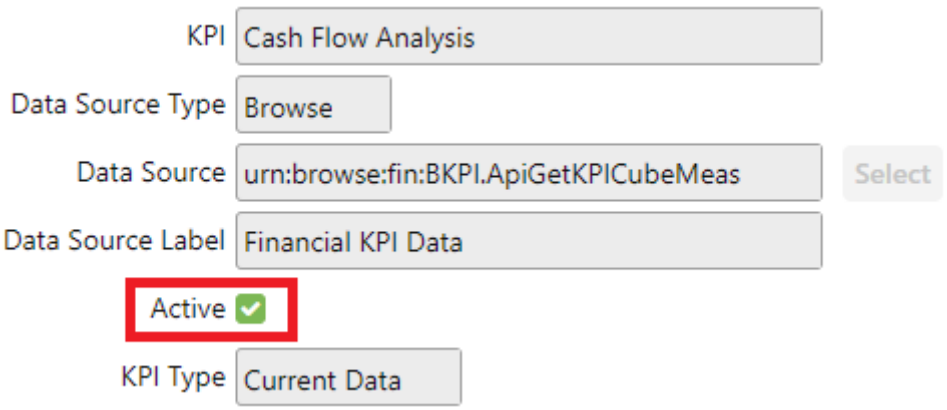
For descriptions of the properties related specifically to Financial Report Writer KPIs, run the command:

```
yab help qad-analytics-financials
```

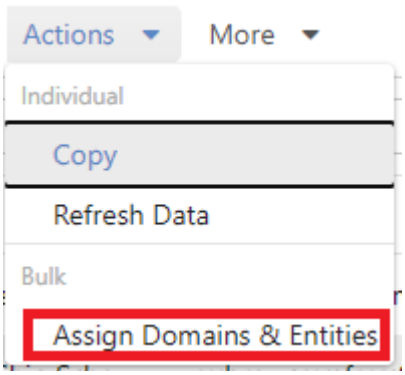
Configuration related to Action Center security is described in Action Center Security.

Activating KPIs

Starting with the September 2020 release, all KPIs have an Active flag. This flag allows KPIs that are not currently being used to be created or imported into the system, so that system resources such as memory and CPU are not consumed unnecessarily to retrieve and cache data for unused KPIs. The feature is especially important for managing KPIs provided by QAD, of which only a subset may be of interest to any one customer. By default, all new KPIs created or installed in an Adaptive UX environment are set to inactive. To enable an unused KPI, select the Active flag on the KPIs screen.



Before the September 2021 release, the Active flag had to be selected manually on the KPIs screen. As of the September 2021 release, a bulk action Assign Domains & Entities has been added to the KPIs browse.



You can use this action to activate any desired set of KPIs, as well as to assign/unassign domain or entity codes to them, without having to update each KPI individually. This action is particularly useful when configuring a new Adaptive UX environment, because until KPIs are activated and assigned valid domains/entities, Action Centers in the environment do not display any data.

KPI Caches

KPIs defined in the Web UI describe the data sets that populate the Action Centers. In order to achieve acceptable performance, they are cached in memory for on-demand display in the Action Centers.

There are several different data sources for KPIs, all of which retrieve data from the OpenEdge databases.

- Browses
- Financial Report Writer (FRW) (obsolete)

As of the March 2022 release, FRW KPIs are no longer supported in Adaptive UX. Only the infrastructure supporting browse-based KPIs will be covered in the current section.

Browse-Based KPIs

KPIs that have browses as their data sources make up the majority of KPIs, because browses use a powerful data retrieval mechanism and can be defined by knowledgeable end users. They have the capability of retrieving data from almost any database table in the system, including financial and operational data.

Pre-defined browse-based KPIs are packaged and installed inside various apps, generally the same apps as the Action Centers that use them.

The browse result sets used by the browse-based KPIs are cached by the Query Service, using its Spark and Cassandra infrastructure.

In Cassandra, browse result sets are persisted to disk in tables that are specific to a combination of browse and entity for financial browses, and browse and domain for all other browses. These tables reside within the 'browses' keyspace. Following are some examples of table names displayed from this Cassandra keyspace.

```
cqlsh> use browses;
cqlsh:browses> describe tables;

kpi_357844472_12mex      kpi_1688405265_20fra     kpi__1355073026_10usa
kpi__846170097_12mex     kpi_1852174264_20fra     kpi__1863008167_22uk
kpi_1988561950_21nl      kpi_1988561950_12mex     kpi__692383936
kpi__1809347128_20fra    kpi__45865388_23ger      kpi__1451599441_10usa
kpi_1202448307_12mex     kpi_1044215493_21nl      kpi_1533800352_30chn
kpi_1202448307_30chn     kpi__1063222905_12mex    kpi_1384866991_10usa
kpi__186238821_12mex     kpi_1705405443_qad       kpi_1988561950_20fra
kpi_1650412229_11can     kpi__1798778746_20fra    kpi__1063222905_11can
kpi__1975819738_20fraco  kpi_1471838795_31aus     kpi_1105371684_10usa
kpi_1206509372_31aus     kpi_1384866991_11can     kpi__1423034166_21nl
kpi__1282164377_23ger    kpi__45865388_31aus      kpi__1809347128_12mex
kpi__1422684978_23ger    kpi__1319591205_22uk     kpi_642442275_10usaco
kpi__1942540096_12mex    kpi__1808255083_10usa    kpi__1604144893_31aus
kpi_1533800352_10usa     kpi__1423034166_11can    kpi__875669433
kpi__78780668_12mexco    kpi_1044215493_31aus     kpi_1529480452_31aus
kpi_1101437247_31aus     kpi_1175672099_12mex     kpi_1852174264_11can
kpi_1854608121_30chn     kpi_1471838795_21nl      kpi_1206509372_11can
kpi__1916630248_11can    kpi_1101437247_40brz     kpi_131336283_21nl
kpi_1080786763           kpi_131336283_22uk       kpi_220484148_12mex
kpi_514817284_10usa      kpi_1529480452_22uk      kpi__6799222_40brz
kpi_1817003066_10usa     kpi_1852174264_80tst     kpi_1533800352_20fra
kpi_1650823036_31aus     kpi__2068936752_30chn    kpi__1809347128_31aus
kpi__1798778746_10usa    kpi__2086541207_23ger    kpi_1721476749_23ger
kpi_1117887640_22uk      kpi_1854608121_40brz     kpi__778172591_10usa
kpi_514817284_11can      kpi_1721476749_40brz     kpi__1918291693_10usa
kpi__1143427273_12mex    kpi_1688405265_21nl      kpi__1537806489_40brz
...

cqlsh:browses>
```

The KPI tables whose names end in a QAD domain or financial entity code (example: "_10usa") contain data from that domain or entity only for a particular KPI. KPI tables with no domain suffix contain data from all domains or entities enabled for that KPI. Unfortunately, the KPI name and its source browse are not identifiable from the table name. Table names may be changed in a future Adaptive UX release to correct this. However, starting with the September 2021 release of AUX, several REST APIs are available that will return the name of the Cassandra tables and Spark views

containing the data for a given KPI. The primary one can be called at the following URL from an HTTP client, such as a web browser or the curl command.

```
<webapp.webshell.url property>/api/analytics/kpiMetadata/table-info?kpiName=<KPI name>
```

Example:

```
https://vmlfwy0000.qad.com:22011/qad-central/api/analytics/kpiMetadata/table-info?
kpiName=Commitments%20and%20Historical%20Spending
```

Note that the KPI name parameter must be URL-encoded to escape the space characters.

Below is sample output from this API request:


```
{
  "errors": [],
  "showResult": true,
  "resultMessage": "",
  "data": {
    "kpiMetadataInfo": {
      "kpiName": "Commitments and Historical Spending",
      "kpiCode": "8a4a88f8-2a59-e6a0-5514-6a2ff8cbf089",
      "kpiType": "Current Data"
    },
    "sparkTableName": "kpi_1721476749___94397eb4dc9aba5faeb22c5347019ecc",
    "cassandraTableNames": [
      "browses.kpi_1721476749_31aus",
      "browses.kpi_1721476749_40brz",
      "browses.kpi_1721476749_11can",
      "browses.kpi_1721476749_30chn",
      "browses.kpi_1721476749_20fra",
      "browses.kpi_1721476749_23ger",
      "browses.kpi_1721476749_12mex",
      "browses.kpi_1721476749_21nl",
      "browses.kpi_1721476749_22uk",
      "browses.kpi_1721476749_10usa"
    ]
  },
  "success": true,
  "errorSeverity": 0
}
```


The Cassandra tables are cached in memory by Spark, which creates views on the fly with appropriate filtering and grouping to support the needs of specific Action Centers.

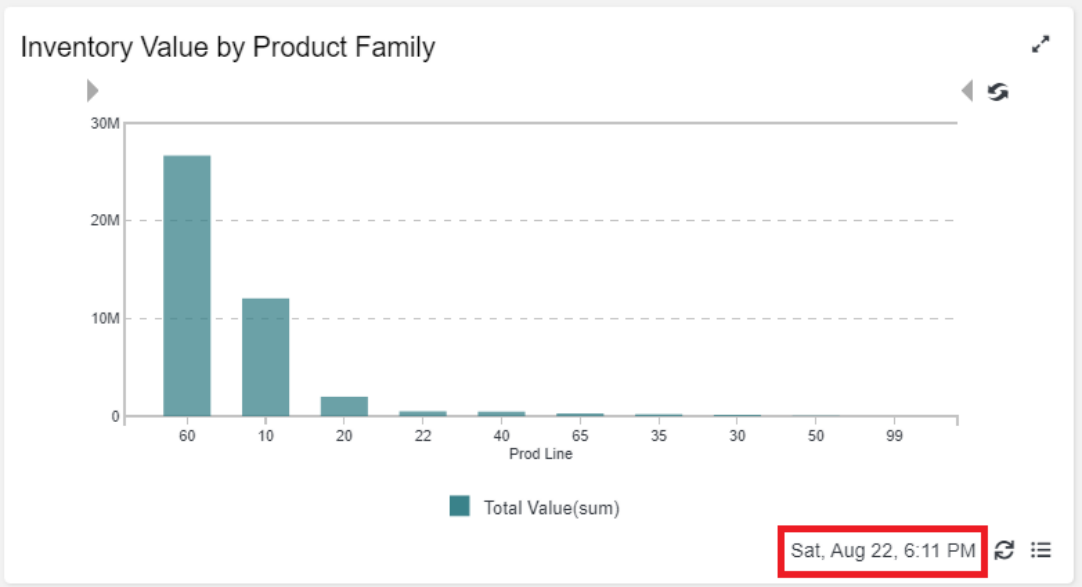
KPI Cache Refreshes

KPI caches can be refreshed with current source data using several configurable approaches:

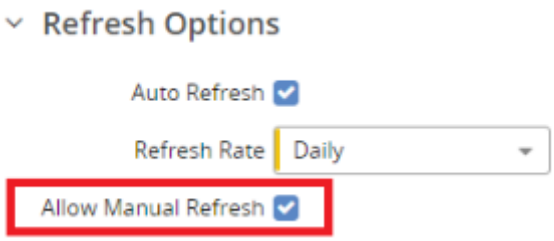
- Manual Refresh
- Cache Warming
- Scheduled Refresh

Manual Refresh

Depending on the configuration of each KPI, it is possible for Action Center users to manually force a refresh of the data from the OpenEdge database by selecting the refresh icon in the lower right corner of a dashboard panel. The icon is displayed next to a date-time stamp showing when the data was last retrieved from the operational database, as highlighted in the following graphic.



The refresh icon is only available if the Allow Manual Refresh flag in the KPI screen is selected for the KPI associated with the panel.



When the refresh icon is selected, the source browse is re-processed, re-cached in memory, and displayed in the Action Center. All the Action Center panels are re-displayed in the browser, but only those panels using the refreshed browse display refreshed source data.

> ⚠ In the case of KPIs whose data source is a large browse, a manual refresh can take several minutes, because the source browse must be processed, the results stored in Cassandra and re-cached in Spark, and the visuals in the Action Center re-rendered by the Logi software.

Because of the potential load in the system when KPIs use large browses, manual refreshes should only be used when current data is required. Routine regular refreshes should be accomplished by enabling *cache warming* and *scheduled refresh*.

The following system-level properties affect KPI caching and refresh behavior, enforcing limits to ensure acceptable online performance.

Property	Purpose/Description	Default Value	Tuning Considerations
qad-analytics-core.maxKpiActiveFields	Maximum number of active fields that may be included in a KPI.	20	Should be limited in order to conserve system resources.
qad-analytics-core.browseBatchRefreshTimeLimit	Causes the copying of a KPI result set into the data lake to be skipped if the same KPI was copied more recently than the value of this property, expressed in minutes.	1	Normally should not be changed.

Cache Warming

Cache warming refers to the process of caching all the KPIs in memory when an environment is started. This allows the KPIs to be available without a significant wait for an Action Center to display the first time one of the underlying KPIs is needed. By default, cache warming is enabled when the system is installed.

In the case of browse-based KPIs cached in the Query Service, it is important to note that cache warming does not necessarily refresh the cache contents to reflect current OpenEdge database contents. If the required browses are already present in the Cassandra data lake, cache warming loads the memory caches from Cassandra without reprocessing the browse requests. Only required browses that are not yet in Cassandra are retrieved from the OpenEdge databases through browse requests. This approach allows cache warming to proceed much faster and with lower system resource usage at application startup, which is often an important practical consideration. Refreshing the caches from the operational database sources is accomplished mainly by scheduled refresh.

Cache warming is controlled by various properties that can be modified using YAB if necessary. This document does not provide a comprehensive list, but only covers those that are most likely to affect overall performance and/or require tuning.

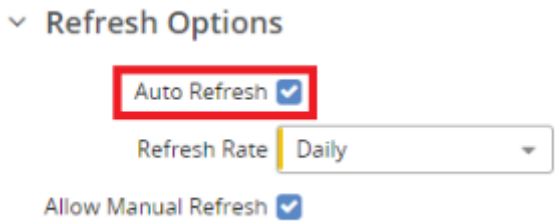
Browse KPI Cache Warming Properties

Property	Purpose /Description	Default Value	Tuning Considerations
`qad-analytics-core.cache.kpis-browse.loadAtStartup`	Indicates if the cache is warmed at application startup.	true	Set to false in order to disable cache warming.
`qad-analytics-core.metricKpiCacheWarmerExecutor.poolSize`	Number of KPI refreshes that can be requested concurrently.	2	Increase to submit refresh requests faster to the Query Service, potentially warming the cache in less time but at the cost of greater resource usage. Must have a value of 1 or greater. The effect of this setting is constrained by the Query Service property qad-qracore.browseCassandraDataService.concurrency (described below).

Scheduled Refresh

Because Action Center displays retrieve their KPI data from in-memory caches, the data may not reflect current database contents. If the data sets are not periodically refreshed, over time they will become stale and less relevant to the needs of the organization. While end users can trigger the data in particular Action Center panels to be refreshed from the OpenEdge sources, manual refreshes are resource intensive and often slow. To keep the Action Center contents current enough to be useful, the system automatically refreshes the browse and FRW KPI caches periodically, based on a configurable schedule. This feature is called 'scheduled refresh.'

Scheduled refresh is configurable by KPI. KPIs can be explicitly enabled for scheduled refresh on a daily, weekly, or monthly basis in the KPI screen using the Auto Refresh setting.



However, there is a limit on the number of KPIs for which scheduled refresh can be enabled, as described below:

Property	Purpose /Description	Default Value	Tuning Considerations
`qad-analytics-core.maxKpisAutoRefreshed`	Maximum number of KPIs for which scheduled refresh may be enabled	30	Can be increased to allow more KPIs to be automatically refreshed, at the cost of more resource-intensive browse requests. The impact of a longer scheduled refresh depends on the number of KPIs, the size of browse result sets, and the overall system load at the time of day when the scheduled refresh is run.

Scheduled refresh is controlled by various properties that can be modified using YAB if necessary. This document does not provide a comprehensive list, but only covers those that are most likely to affect overall performance and/or require tuning.

The scheduled refresh process is started and runs inside the tomcat-webui instance, not in separate scripts. If tomcat-webui is not running at the time when the scheduled refresh is scheduled (ex. during an offline backup), or was stopped before an in-process scheduled refresh could complete, no special recovery process is initiated. Instead, the scheduled refresh will run at the next scheduled date-time once tomcat-webui is running again.

Browse KPI Scheduled Refresh Properties

Property	Purpose/Description	Default Value	Tuning Considerations
`qad-analytics-core.cache.kpis-browse.scheduledRefresh.enabled`	Enables scheduled refresh processing across the system, based on the other properties and individual KPI configuration.	true	Set to false to disable all scheduled refreshes.
`qad-analytics-core.cache.kpis-browse.scheduledRefresh.batchSize`	Maximum number of requests that are batched for processing by a single KPI refresh thread.	100	Lower values may allow the scheduled refresh to be completed faster, at the cost of using more memory and/or processor resources. Not recommended to change.
`qad-analytics-core.cache.kpis-browse.scheduledRefresh.quartzJobCount`	Number of background threads that can concurrently request KPI refreshes. Must be greater than or equal to 2.	2	More threads would allow more scheduled refreshes to run concurrently, at the cost of using more memory and/or processor resources.
`qad-analytics-core.cache.kpis-browse.scheduledRefresh.cronExpression`	String expression in a cron format that specifies schedule when the scheduled refresh is run. See the Quartz documentation for a more detailed description of cron syntax. This setting does not cause cron scripts to run. The cron syntax is only used to set the schedule for background refresh activity run within the Tomcat environment.	0 0 0 * * ? (every day at midnight)	Set to a time schedule that suits the organization, based on system load and user activity over a 24-hour period. If possible, schedule for a time of day with low online user activity.
`qad-analytics-core.cache.kpis-browse.scheduledRefresh.kpi-weekly-day`	For KPIs that are configured to be refreshed weekly, specifies the day of the week on which the refresh is performed. Valid values are sun, mon, tue, wed, thu, fri, sat.	sun	Set to a day of the week that suits the organization, based on system load and user activity.
`qad-analytics-core.cache.kpis-browse.scheduledRefresh.kpi-monthly-day`	For KPIs that are configured to be refreshed monthly, specifies the day of the month on which the refresh is performed. Syntax is a comma-separated string with two values: • first or last: Indicates if the refresh is defined relative to the beginning or end of each month. • offset days: Sets the number of days before or after the first or last of the month to perform the refresh. A positive value states the number of days after the first or last day of the month; a negative value states the number of days before the first or last day of the month; and a value of 0 indicates the first or last day of the month with no offset.	first,0	Set to a day of the month that suits the organization, based on system load and user activity.
`qad-analytics-core.browseBatchRefreshTimeLimit`	Minimum number of minutes allowed between refreshes of the same KPIs browse by the Query Service. If a requested browse was refreshed within this time interval, the new request is skipped. This property prevents repeated refreshes of the same browse from being processed within a short time interval (for example, by repeatedly clicking the refresh control inside an Action Center panel).	1	Increase the value in order to increase the minimum amount of time allowed for refreshes of the same browse. This may be important in order to prevent excessive load on the system, in terms of AppServer agents and SQL connections that are consumed processing browse requests. Smaller values allow the same browse to be refreshed more frequently. This property allows you to adjust the trade-off between system resource usage and data currency of Action Center displays.

Key Query Service Properties

The following properties are specific to the Query Service, and therefore affect the caching of only browse-based KPIs, not FRW KPIs. They can affect both cache warming and scheduled refresh processing. This is not a comprehensive list, but only covers those properties that are most likely to affect overall performance and/or require tuning.

Property	Purpose /Description	Default Value	Tuning Considerations
`qad-qracore.browseCassandraDataService.concurrency`	Number of browse retrievals and data copies into Cassandra that can be processed concurrently.	3	Limits the number of concurrent browse requests that can be processed by the Query Service. Browse requests are one of the following types. • SQL queries using an OpenEdge JDBC connection • Browse engine queries run on a QRA AppServer agent. This property limits the number of browse requests of either type that can be processed at the same time. For browse requests processed on the QRA AppServer, this is a critical setting, as Progress AppServers are often a scarce and CPU-intensive system resource. If the number of AppServer agents being used to process Query Service browse requests causes the environment to run out of available agents, a fatal 'No Servers Available' error may be raised by Progress. This error will cause the request that needs the AppServer agent to fail, whether it comes from the Query Service or from another system component. For this reason, this property should be set to a value lower than the total number of QRA AppServer agents available in the environment, allowing enough agents available for other activities to proceed while cache warming or a scheduled refresh is in process.
`qad-qracore.browseCassandraDataService.timeoutLimit`	Maximum number of seconds that the Query Service will wait for a page of output to be returned from a browse request.	120	In a heavily loaded environment with high CPU and/or data retrieval activity, especially using AppServer agents, this setting might have to be increased to give the system time to retrieve complete browse results.
`qad-qracore.browseCassandraDataService.pageSize`	Number of records retrieved in a single page or chunk from a browse request.	5000	Could be adjusted to retrieve browse data from the source in smaller or larger chunks, potentially affecting data retrieval traffic and latency.

Monitoring Cache Refreshes

There is no console or screen to monitor the progress of cache warming or scheduled refreshes. However, a portion of the Query Service activity can be viewed as a list of jobs shown in the Spark web UI, briefly introduced in Action Center Maintenance and Troubleshooting. In addition, progress messages are written to the tomcat-webui console log when the appropriate settings are configured in the `logback.xml` file. Instructions for configuring logging, either manually or through YAB, are not covered in this document. However, the settings necessary in logback.xml to obtain comprehensive KPI caching messages are summarized below.

```
    <logger name="com.qad.analytics.core.service.impl.KpiCacheLoaderBase" level="debug"
additivity="false">
        <appender-ref ref="stdout" />
    </logger>
    <logger name="com.qad.analytics.core.service.impl.BrowseKpiCacheLoader" level="debug"
additivity="false">
        <appender-ref ref="stdout" />
    </logger>
    <logger name="com.qad.analytics.financials.service.impl.FrwKpiCacheLoader2" level="debug"
additivity="false">
        <appender-ref ref="stdout" />
    </logger>
```

For these settings, it is assumed that the appender *stdout* references the standard tomcat-webui console log file. The resulting messages have a log level of DEBUG. They trace the caching of each KPI for each domain or financial entity, without showing details about the related browses or queries.

View-As User Feature

> The View-As User feature is not available in AUX environments that use Logi Composer, starting with the September 2022 release.

Starting with the March 2020 release, it is possible for a Web UI user to display an Action Center in the same way that it would be displayed to a different Web UI user who is subordinate in the organization to the actual user. This feature allows a manager, for example, to easily see the same Action Center data that one of his or her subordinates would see. Special configuration in the Web UI is required to enable this feature in a secure way. The set up and use of the feature is described in the online help and Adaptive UX User Guide.

However, using View-As User can decrease the online performance of the Action Centers and requires additional security configuration. In addition, in many customer environments the feature will not be used at all. Therefore, it has been disabled by default at the system level using the following Web UI property.

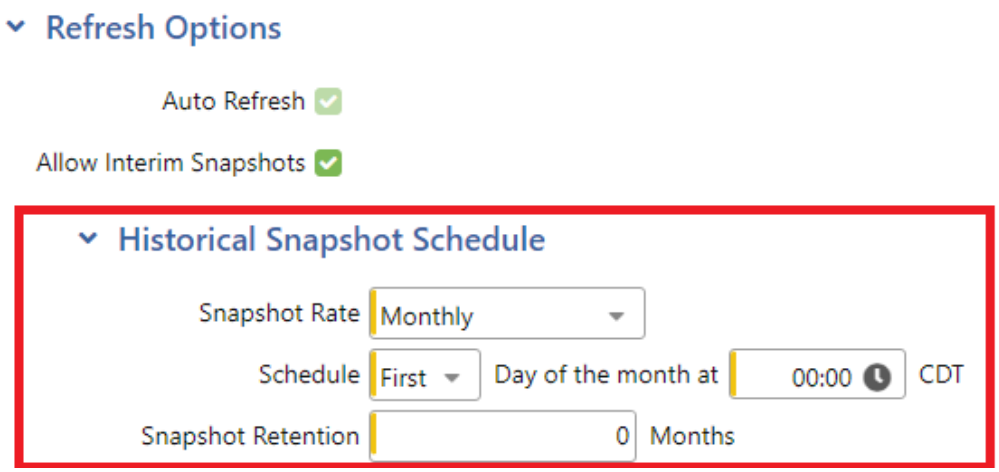
```
qad-analytics-core.logips.viewAs.enabled=false
```

In order to configure and use 'View-As User' permissions in the Web UI, the feature must first be enabled. To do this, set the property to `true` using YAB. It will be activated after the next YAB update and tomcat-webui restart.

Historical KPIs Feature

Starting with the September 2020 release, historical KPIs were introduced. Historical KPIs allow you to track data trends by comparing summary-level metrics over time. In the KPIs view, you can create historical KPIs and configure the system to take snapshots of your KPI for particular periods. You can then compare these periods to analyze the data.

Like the Scheduled and Manual Refresh functions, historical snapshots are configured by KPI on the KPI screen.



A chronological list of existing snapshots is displayed in the Snapshot History section of the KPI screen, for all historical KPIs.

However, there are limits on the number and size of snapshots that are configured at the system level, as described below.

> Because historical KPI snapshots accumulate over time based on their frequencies and schedules, they can grow to consume large amounts of disk space depending on their size, grouping level, and so on. The limits on historical KPI snapshots should therefore be considered carefully for each Adaptive UX installation.

Historical KPI Properties

Property	Purpose/Description	Default Value	Tuning Considerations
`qad-analytics-core.kpi.historical.`	Determines if historical KPI functionality is enabled.	true	

enabled			
qad-analytics-core.kpi.historical.schedule.enabled	Indicates if the historical KPI schedule is enabled. This property depends on qad-analytics-core.kpi.historical.enabled and if it is disabled, then the historical KPI schedule is also disabled despite the value of the property.	true	
qad-analytics-core.kpi.historical.maxSnapshots	Maximum number of snapshots in the system across all KPIs, regardless of snapshot frequency or size. Once the limit is reached, the oldest snapshot is automatically deleted.	1100	
qad-analytics-core.kpi.historical.maxRowCount	Maximum number of records that may be retrieved for a single Historical KPI snapshot.	200000	
qad-analytics-core.kpi.historical.maxKpiActiveFields	Maximum number of fields or data columns that may be enabled for a single Historical KPI.	20	
qad-analytics-core.kpi.historical.cassandra.gc_grace_seconds	Delay of final deletion of tombstones in Historical Data tables in Cassandra.	0	A value of zero causes tombstones to be deleted immediately, which is good in a single-node cluster. However, in a multi-node cluster a value of zero would cause data integrity problems across the nodes in the cluster.

Special Logi Info Configuration

As of the September 2021 release, Logi Info is not supported and KPIs created using Logi Info can no longer be displayed.

In releases prior to September 2019 when all Action Centers were supported using Logi Info, there were several situations that required manual configuration changes to the Logi Info settings. The following information is not necessary for environments created with the September 2019 release or later.

Maximum Number of Data Rows

When KPI are cached that return a large number of data rows through the processing of browse requests, significant work loads can be added to the computer system, especially additional CPU usage. In some cases, the extra load can affect system stability and degrade the response times for all users. Similar issues can sometimes be seen when running large QAD browses outside of an Action Center context.

To address this potential issue, a system-level property is provided to limit the number of data rows that can be returned for a single KPI and workspace.

Property	Purpose/Description	Default Value	Tuning Considerations
qad-analytics-core.maxRowCount	Maximum number of browse rows that can be retrieved for any KPI in the system for a single domain or entity.	5000	Raise only after considering the additional system load that will be incurred by processing large browses in order to retrieve KPI data.

Special Calendar Support

Fiscal Year and Quarter

For organizations that use a fiscal calendar different from the standard calendar, Action Centers in Logi Info allow time-line charts to be created that present dates in terms of fiscal year or fiscal quarter, as well as calendar year or quarter. However, the fiscal periods used by Logi Info are not integrated with the fiscal calendar used in Adaptive Applications. In order to see the Fiscal Year and Fiscal Quarter options when configuring visuals for Action Centers, you must manually add a property to the main Logi Info configuration file.

The name of this file is `_Settings.lgx`. It is saved in the `_Definitions/ directory` under the root directory of the qad-dashboards webapp. To find this root directory, run the following command.

```
yab config webapp.analytics-logi.dir
```

Open the file with a text editor. Find the Globalization element near the end of the file:

```
<?xml version="1.0" encoding="utf-8"?>
<Setting ID="_settings">
    <GlobalCSS StyleSheet="QAD.Qracore.actioncenter.css" Theme="ChannelIslands"/>
    <Application/>
...
    <Globalization UserCulture="@Session.UserDisplayLocale~"/>
    <ideTestParams/>
</Setting>
```

Add the attribute `FirstDayOfFiscalYear` to the Globalization element as shown below. Set its value to MM/DD, where MM is the two-digit month of the calendar year and DD is the two-digit day of the month.

```
<?xml version="1.0" encoding="utf-8"?>
<Setting ID="_settings">
    <GlobalCSS StyleSheet="QAD.Qracore.actioncenter.css" Theme="ChannelIslands"/>
    <Application/>
...
    <Globalization FirstDayOfFiscalYear="02/01" UserCulture="@Session.UserDisplayLocale~"/>
    <ideTestParams/>
</Setting>
```

Save the change. It will take effect immediately, with no need to restart the environment.

If this property is not present in the file, Logi Info assumes that the fiscal and calendar years are the same, with no need for the Fiscal Year or Fiscal Quarter options when configuring visuals.

Week Start Day

Action Centers in Logi Info allow time-line charts to be created that present dates in one-week blocks, by selecting the Week option when configuring the chart. However, some organizations prefer to express weeks with non-standard start and end days, rather than Sunday through Saturday. In order to set the start day of the week for purposes of creating visuals, you must manually add a property to the main Logi Info configuration file.

The name of this file is `_Settings.lgx`. It is saved in the `_Definitions/ directory` under the root directory of the qad-dashboards webapp. To find this root directory, run the following command:

```
yab config webapp.analytics-logi.dir
```

Open the file with a text editor. Find the Globalization element near the end of the file:

```
<?xml version="1.0" encoding="utf-8"?>
<Setting ID="_settings">
    <GlobalCSS StyleSheet="QAD.Qracore.actioncenter.css" Theme="ChannelIslands"/>
    <Application/>
...
    <Globalization UserCulture="@Session.UserDisplayLocale~"/>
    <ideTestParams/>
</Setting>
```

Add the attribute `FirstDayOfWeek` to the Globalization element as shown below. Set its value to one of the following:

- 0 = Sunday (the default value)
- 1 = Monday
- 2 = Tuesday
- 3 = Wednesday
- 4 = Thursday
- 5 = Friday
- 6 = Saturday

```
<?xml version="1.0" encoding="utf-8"?>
<Setting ID="_settings">
    <GlobalCSS StyleSheet="QAD.Qracore.actioncenter.css" Theme="ChannelIslands"/>
    <Application/>
...
    <Globalization FirstDayOfWeek="1" UserCulture="@Session.UserDisplayLocale~"/>
```

```
    <ideTestParams/>
</Setting>
```

Save the change. It will take effect immediately, with no need to restart the environment.

Action Center Maintenance and Troubleshooting

Backup and Restore Activities

This section summarizes considerations for regular data backup and restore activities that are specific to Action Centers and the Query Service.

Logi Composer (Zoomdata) Databases

Starting with the September 2022 release, the Action Center dashboard and visual definitions developed using Logi Composer are stored in PostgreSQL databases that are installed with Composer. In Composer-enabled environments, the H2 database embedded as part of Logi Platform Services is not used.

The databases, managed within the same PostgreSQL instance, are named zoomdata and zoomdata-qe. Their contents are changed online as a result of end-user activity that creates, modifies, or deletes KPIs, Action Centers, and visuals. They also contains much configuration data internal to Composer. As the PostgreSQL activity is not recorded in OpenEdge database rollback or roll-forward logs, their contents cannot be precisely synchronized with the OpenEdge databases when the databases must be restored to a particular point in time through automatic roll-forward operations, such as in some disaster recovery scenarios. However, in practice the risk of data corruption is low, given the following points.

- KPI, Action Center, and visuals maintenance are typically low-volume, low-frequency activities that are performed by only a subset of users during working hours.
- Most of the contents of are not tightly coupled with OpenEdge database contents. In addition, the restore process run by YAB also calls a REST API in AUX that ensures Composer contains the same objects with the same identifiers as the QAD databases. As a result, changes to the PostgreSQL tables generally do not affect the QAD databases and vice versa.

To minimize the risk of data loss, the PostgreSQL databases should always be backed up at the same time as the OpenEdge databases. The following YAB command creates a backup of the PostgreSQL contents:

```
yab postgresql-default-backup
```

The backups are stored in the directory referenced by the following YAB property:

```
postgresql._backup.dir
```

The following YAB commands include the PostgreSQL databases in backups of the system environment:

```
yab environment-online-backup
yab environment-offline-backup
```

The following YAB command provides a list of all the existing PostgreSQL backups for Logi Composer:

```
yab postgresql-default-backup-list
```

The output of the backup list command is similar to the following example:

```
Tag: default
---------------
Location: /dr01/dbs/backup/default/postgresql

default                                 Jul 11, 2022 9:59:22 AM

Tag: 20220711120330
---------------
Location: /dr01/dbs/backup/20220711120330/postgresql

default                                 Jul 11, 2022 10:03:30 AM
```

The following YAB command deletes all Logi Composer PostgreSQL backups:

```
yab postgresql-backup-remove
```

It is executed as part of the following YAB command:

```
yab database-all-backup-remove
```

PostgreSQL backups can be restored only when Logi Composer is offline. They are restored by the following YAB command:

```
yab postgresql-default-restore
```

The restore is also executed as part of the following YAB command:

```
yab database-all-restore
```

Logi Platform Services Product Database

Starting with the September 2022 release, Logi Composer is the only option available to install in new AUX environments and the default option for existing AUX environments. While Logi Platform Services can still be used in older AUX environments that have been upgraded to September 2022, it is deprecated and will be retired in a future AUX release.

Starting with the September 2019 release, the Action Center dashboard and visual definitions developed using Logi Platform Services are stored in Logi's Product Database (PDB), which is implemented using the H2 relational database manager. Its contents are changed online as a result of end-user activity that creates, modifies, or deletes KPIs, Action Centers, and visuals. The PDB also contains much configuration data internal to LogiPS. PDB activity is not recorded in OpenEdge database rollback or roll-forward logs. As a result, the PDB cannot be precisely synchronized with the OpenEdge databases when the databases must be restored to a particular point in time through automatic roll-forward operations, such as in some disaster recovery scenarios. In practice, the risk of data corruption is relatively low, given the following points.

- KPI, Action Center, and visuals maintenance are typically low-volume, low-frequency activities that are performed by only a subset of users during working hours.
- The contents of the PDB are not tightly coupled with OpenEdge database contents. As a result, changes to the Logi tables generally do not affect the QAD databases and vice versa.

To minimize the risk of data loss, the PDB should always be backed up at the same time as the OpenEdge databases. The following YAB command creates a backup of the Logi Platform Services PDB.

```
yab logi-platform-services-default-backup
```

The backups are stored in the directory referenced by the following YAB property:

```
logi-platform-services-backup.dir
```

The following YAB commands include the PDB in backups of the system environment:

```
yab environment-online-backup
yab environment-offline-backup
```

The following YAB command provides a list of all the existing PDB backups for Logi Platform Services:

```
logi-platform-services-default-backup-list
```

The output of the backup list command is similar to the following example:

```
Tag: default
--------------
Location: /dr01/dbs/backup/default/logi-platform-services

default.zip                          May 27, 2019 8:05:27 AM

Tag: 20190527080605
--------------
Location: /dr01/dbs/backup/20190527080605/logi-platform-services

default.zip                          May 27, 2019 8:06:06 AM

Tag: foo
--------------
Location: /dr01/dbs/backup/foo/logi-platform-services

default.zip                           May 27, 2019 8:04:40 AM
```

The following YAB command deletes all Logi Platform Services PDB backups:

```
yab logi-platform-services-backup-remove
```

It is executed as part of the following YAB command:

```
yab database-all-backup-remove
```

The PDB can be restored only when Logi Platform Services is offline. It is restored by the following YAB command:

```
yab logi-platform-services-default-restore
```

The restore is also executed as part of the following YAB command:

```
yab environment-restore
```

Because the PDB contains license information, if a PDB backup from an environment containing a different Logi Platform Services license is restored into the target environment, the correct license should be re-imported after the restore using the following YAB command:

```
yab logi-platform-services-default-license-import
```

Logi Files

As of the September 2021 release, Logi Info is not supported and KPIs created using Logi Info can no longer be displayed.

Prior to the September 2019 release, the Action Center dashboard and visual definitions were stored in XML files inside the Logi Info web application, not in a database. For environments still running a pre-September 2019 release, these files are changed online as a result of end-user activity that creates, modifies, or deletes KPIs, Action Centers, and visuals. Changes to these files are not recorded in OpenEdge database rollback or roll-forward logs. As a result, these files cannot be precisely synchronized with the OpenEdge databases when the databases must be restored to a particular point in time through automatic roll-forward operations, such as in some disaster recovery scenarios. In practice, the risk of data corruption is relatively low, given the following points:

- KPI, Action Center, and visuals maintenance are typically low-volume, low-frequency activities that are performed by only a subset of users during working hours.
- The contents of the Logi files are not tightly coupled with OpenEdge database contents. As a result, changes to the Logi files generally do not affect the database and vice versa.

All the Action Center files are stored in a single directory inside the Logi Info web app. To find the directory location, query the YAB configuration:

```
yab config qad-analytics-core.logiDashboardsPath
```

To minimize the risk of data loss, the Logi files should always be backed up at the same time as the OpenEdge databases. The following YAB commands include the Logi files in backups of the system environment:

```
yab environment-online-backup
yab environment-offline-backup
yab directorybackup-backup
yab directory-action-center-backup
```

Data restore activities performed by system administration personnel should be implemented to include both the OpenEdge databases and the Logi files. The files can be backed up and restored through simple file copies, without the use of any special utilities. The following YAB commands restore the Logi files backups created by a previous YAB backup:

```
yab directorybackup-restore
yab directory-action-center-restore
```

Because changes to the Logi files are not automatically logged, QAD recommends that the Logi files be backed up more frequently than the OpenEdge databases in order to support recovery procedures when it is necessary to restore the entire system to a stable state as of a particular point in time. If the OpenEdge databases must be restored and rolled forward to a particular point in time, more frequent Logi file backups allow the files to be restored to a state that is closer to the restored database state. There is still the possibility of some Action Center data loss if the current Logi files were lost during a severe service outage, but the risk can usually be managed to a low level through this approach.

Cassandra Keyspaces

The 'browses' keyspace in the Cassandra data lake that contains Query Service data is not a system of record, but is created entirely from the contents of operational OpenEdge database tables. There is no need to back up its contents or restore a backup in case of data loss. It should therefore be included in the list of keyspaces exempted from backups in the YAB property cassandra.default.node.backup.blacklist. Whenever there is a need to restore/refresh the data in this keyspace to the current state, the keyspace should be rebuilt from its sources as described in the section 'Rebuilding the Cassandra Keyspace.'

The 'historical_kpi' keyspace in the Cassandra data lake is the system of record for historical KPI snapshots, and must be included in all database backups. It should therefore be omitted from the list of keyspaces exempted from backups in the YAB property cassandra.default.node.backup.blacklist.

Spark Cache

Within the Query Service, Spark is used as an on-demand, memory-based cache of browse data that is loaded from the Cassandra data lake. Hence, there is no need to back up its contents or restore a backup in case of data loss. Instead, the cache is rebuilt or refreshed at the same time as the Cassandra keyspace (see above).

Log Files

Because much of the Action Center and Query Service processing is performed in the background and is not visible in the user interface, log files are the most important resource for diagnosing problems.

Tomcat Logs

The console log file written by the Tomcat instance that supports the Web UI (tomcat-webui) is the first place to look for error details. By default, the current log file is named catalina.out, and the files created on previous days are named catalina.<date>.log, where <date> is the date when the log was written. These file are stored in the logs/directory under the Tomcat instance. To find the root directory of the Tomcat instance, run the following command.

```
yab config tomcat.webui.base
```

Logi Composer Logs

Logi Composer logs are stored in several locations in the system. To find the primary log file location, query the YAB configuration as follows:

```
yab config logi-composer.default.logs
```

Various Composer microservice and data connector components also write separate log files. While they can be configured to write to different locations in the file system, by default they write to the same location as the one mentioned above. In case different locations are used, the YAB configuration can be queried using several commands:

```
yab config logi-composer.default.microservice.zoomdata.configuration.logs.dir
yab config logi-composer.default.microservice.zoomdata-query-engine.configuration.logs.dir
yab config logi-composer.default.microservice.zoomdata-edc-postgresql.configuration.logs.dir
```

PostgreSQL database logs are written to a different location, queried as shown below:

```
yab config postgresql.default.config.log_directory
```

Logi Platform Services Logs

To find the Logi Platform Services log files generated by the Logi Data Service, query the YAB configuration:

```
yab config logi-platform-services.default.log.dir
```

The log files in this location are as follows:

- `lps-default-dataservice.log`: describes starting and stopping DataServices and H2 services, and displays errors with the services.
- `lps-default-dataservice-error.log`: describes things like JWS errors and other service errors.

To find the Logi Platform Services log files generated by the Logi Application Service, query the following property and go to the logs/ sub-directory under it.

```
yab config logi-platform-services.default.dir
```

The log files in this location are as follows:

- `logiApplicationService.<date>.log`: describes possible connection errors.
- `logiDataServiceStartup.log`: describes backup locations and related java information.
- `microservice-<date>.log`: describes errors in Logi's microservice processes (example: PDF export actions).
- `licenseImport log`: describes the license import processes.
- `clientSecret.log`: describes the process that assigned client secret codes.

Cassandra Logs

To find the Cassandra log files, query the YAB configuration.

```
yab config cassandra.default.node.jvm.logs.dir
```

The log files are located in the sub-directory default/ within this directory. The `cassandra-default.log` file shows Cassandra activity, and the `gc.log.*` files show its Java garbage collection activity.

Spark Logs

To find the Spark log files, query the YAB configuration for the master and slave processes running in Spark:

```
yab config spark.masterdefault.env.spark.log.dir
```

The Spark master process manages the resource used by Spark workers to process particular requests:

```
yab config spark.slavedefault.env.spark.log.dir
```

Spark worker processes carry out particular tasks based on the incoming requests.

YAB Logs

If errors related to Action Centers or the Query Service are raised while running a YAB command, consult the YAB log file for details recorded by YAB (for example, a cache warming failure when the Tomcat instance is started during a YAB update).

The YAB log file is named yab.log, and is stored in the build/logs/ directory under the root of the installation.

Repair and Refresh APIs

The APIs described in this section should be run by system admin personnel only, as they can take a long time to run and consume significant system resources.

Logi Composer Sync API

Starting with the September 2022 release when Composer is introduced, a REST API and associated YAB command are provided to synchronize the Action Centers, KPIs, and visuals stored in AUX with the corresponding objects stored in Logi Composer. This API is run automatically following the restore of a PostgreSQL database backup and after migrating Logi Platform Services artifacts to Logi Composer during a YAB update.

To call this API directly, run the following YAB command:

```
yab webapp-analytics-composer-api-configure
```

The API can also be called from a web browser or other HTTP client tool at the following endpoint:

HTTP GET /api/analytics/composer/sync

Logi Platform Services Repair APIs

Starting with the September 2019 release, REST APIs are available to help solve Adaptive UX environment problems caused by the Action Center data in Adaptive UX and Logi Platform Services being out of sync. This can occur when a YAB create or YAB update fails due to various causes in the environment configuration or in specific KPIs. In such cases, it is often difficult to trace the root cause of the problem, especially when different YAB commands have been run multiple times with different failures.

Repair Logi Platform Services Action Centers

It is possible that a KPI may be present in the environment without the Logi Platform Service dataview object that is required to support it. In this case, an Action Center display may contain empty panels with no specific error messages, but show the message 'Data retrieval in process' indefinitely in the lower right corner of the panel. The `catalina.out` log file for tomcat-webui contains an error 'Table or view not found,' as shown in the following example.

```
[19/08/13@04:34:38.943-0700] WARN    org.apache.hadoop.hive.metastore.ObjectStore: Failed to get
database qad_global_temp, returning NoSuchObjectException
[19/08/13@04:34:38.957-0700] ERROR   org.apache.spark.sql.hive.thriftserver.
SparkExecuteStatementOperation: Error executing query, currentState RUNNING,
org.apache.spark.sql.AnalysisException: Table or view not found: `qad_global_temp`.
`mfg_ic761___70c908255fec4a2d66002148fd41a2e1`; line 1 pos 1238;
'GlobalLimit 1000
+- 'LocalLimit 1000
   +- 'Project ['Query.in_mstr__in_cnt_date AS cnt_date#60445, 'Query.in_mstr__in_domain AS
domain#60446, 'Query.in_mstr__in_iss_date AS iss_date#60447, 'Query.in_mstr__in_part AS
part#60448, 'Query.in_mstr__in_qty_oh AS qty_oh#60449, 'Query.in_mstr__in_rec_date AS
rec_date#60450, 'Query.in_mstr__in_site AS site#60451, 'Query.local_variables__local_var02 AS
ABC_Class#60452, 'Query.local_variables__local_var08 AS Nettable_Value#60453, 'Query.
pt_mstr__pt_abc_qty AS abc_qty#60454, 'Query.pt_mstr__pt_desc1 AS desc1#60455, 'Query.
pt_mstr__pt_group AS group#60456, 'Query.pt_mstr__pt_part_type AS part_type#60457, 'Query.
pt_mstr__pt_prod_line AS prod_line#60458, 'Query.pt_mstr__pt_status AS status#60459]
      +- 'Filter (('Query.domainCode = 10USA) && ('Query.domainCode = 10USA))
         +- 'SubqueryAlias Query
            +- 'Project ['to_utc_timestamp('in_mstr__in_cnt_date, America/Los_Angeles) AS
in_mstr__in_cnt_date#60442, 'in_mstr__in_domain, 'to_utc_timestamp('in_mstr__in_iss_date, America
/Los_Angeles) AS in_mstr__in_iss_date#60443, 'in_mstr__in_part, 'in_mstr__in_qty_oh,
'to_utc_timestamp('in_mstr__in_rec_date, America/Los_Angeles) AS in_mstr__in_rec_date#60444,
'in_mstr__in_site, 'local_variables__local_var02, 'local_variables__local_var08,
'pt_mstr__pt_abc_qty, 'pt_mstr__pt_desc1, 'pt_mstr__pt_group, 'pt_mstr__pt_part_type,
'pt_mstr__pt_prod_line, 'pt_mstr__pt_status, 'domainCode]
               +- 'UnresolvedRelation `qad_global_temp`.
```

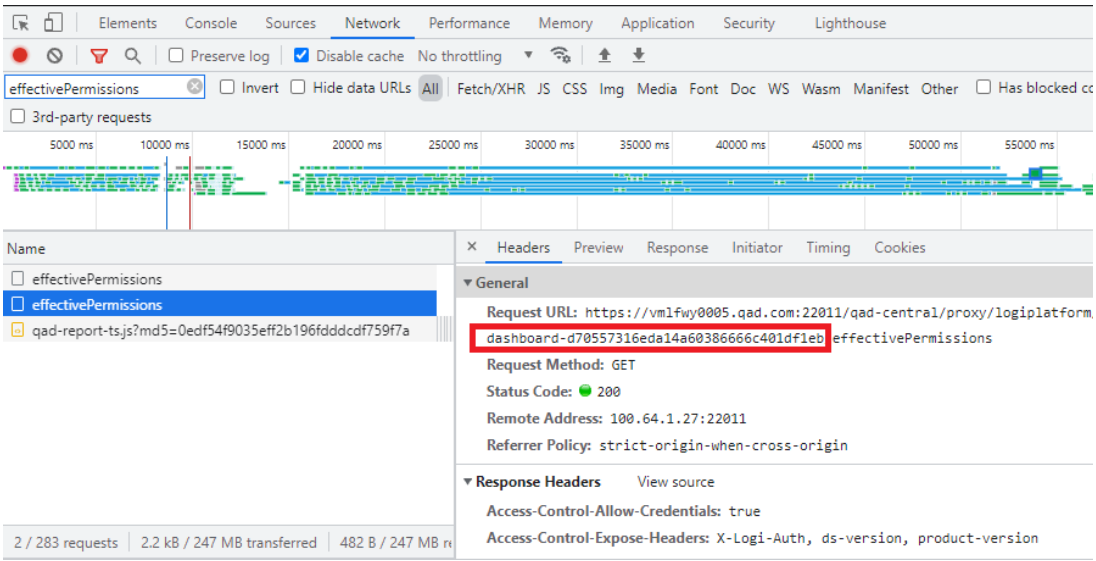
```
`mfg_ic761___70c908255fec4a2d66002148fd41a2e1`        at org.apache.spark.sql.catalyst.analysis.
package$AnalysisErrorAt.failAnalysis(package.scala:42)
        at org.apache.spark.sql.catalyst.analysis.CheckAnalysis$$anonfun$checkAnalysis$1.apply
(CheckAnalysis.scala:82)
        at org.apache.spark.sql.catalyst.analysis.CheckAnalysis$$anonfun$checkAnalysis$1.apply
(CheckAnalysis.scala:78)
...
```

In this case, running the following REST API from the web browser or another HTTP client tool may fix the problem by adding/updating the Logi dataviews used by the Action Center.

HTTP GET /api/analytics/lps/repair/dashboard?dashboardid=<dashboard ID>

To find the dashboard ID value for a given Action Center:

1. Press the F12 key in the web browser to open the DevTools console.
2. Go to the Network tab.
3. Refresh the browser page while the Action Center is displayed.
4. Type the string "effectivePermissions" into the Filter box of DevTools.
5. Find the URL containing the string "/system.configs/dashboard-". The dashboard ID is a string prefixed with 'dashboard-', as in the following example.



Beginning with the March 2020 release, all Action Centers in the system along with all their associated Logi artifacts can be repaired by running the following YAB command, which can take many minutes to run

```
yab action-center-dashboard-repair
```

This command is automatically run during a yab create or yab update.

Delete Orphaned Logi Platform Services Objects

Sometimes problems encountered during the creation of new environments or the export-import of Action Centers and KPIs into existing environments can cause the contents of the QAD and Logi databases to become out of sync in several ways.

- Dashboards (Action Centers) in the Logi database do not have associated records in the QAD database.
- Dataviews in the Logi database do not have associated KPIs in the QAD database.
- App or KPI tags exist in the Logi database without an associated app or KPI in the QAD database.

Often these cases will not affect Adaptive UX operation, as 'orphaned' objects in the Logi database will usually not be accessible or visible to QAD users. However, occasionally special technical problems may result. As of the September 2021 release, the following REST API can be run in the web browser or other HTTP client tool to delete these objects:

HTTP GET /api/analytics/cleanup/lps/corrupted

It is intended to be used only by Support personnel and should not be run routinely.

KPI Refresh

When it is necessary to refresh the KPI data stored in Cassandra and cached in Spark, manual refreshes can be performed for a single KPI by selection Actions > Refresh Data on the KPI screen. However, occasionally there is the need to force a refresh of the data displayed by all KPIs without waiting for the next scheduled refresh run. You can do this by running the following REST API in a web browser or other HTTP client tool:

HTTP GET /api/analytics/lps/refresh/kpis

Depending on the environment, this API can take a lot of time and consume significant system resources. You should avoid running this API in production environments.

KPI Table Information API

While the datasets for all KPIs are stored in Cassandra and cached in Spark, the names of the Cassandra tables and Spark views associated with each KPI do not reflect the human-readable KPI names. Instead, hash codes are generated to name the tables and views, to ensure that all KPIs are unique and meet the naming constraints of both Cassandra and Spark. Therefore, when trying to query the Cassandra table or Spark view containing the data for a particular KPI, it is difficult to find the table/view name for the KPI of interest.

To address this problem, the following REST API, which was introduced in September 2021, returns information about where KPIs are stored in Cassandra and Spark.

HTTP GET /api/analytics/kpiMetadata/table-info?kpiName=<KPI name>

When run from the web browser or another HTTP client tool, it returns JSON data for the requested KPI, as in the following example for the KPI "Top Customers."


```
{
    "errors": [],
    "showResult": true,
    "resultMessage": "",
    "data": {
        "kpiMetadataInfo": {
            "kpiName": "Top Items",
            "kpiCode": "994dcb3b-d29c-bf89-5614-675f4010b8b2",
            "kpiType": "Current Data"
        },
        "sparkTableName": "kpi_1817003066___afd7af965b156ba801a896a0995dd3a6",
        "cassandraTableNames": [
            "browses.kpi_1817003066_10usa",
            "browses.kpi_1817003066_12mex",
            "browses.kpi_1817003066_11can"
        ]
    },
    "success": true,
    "errorSeverity": 0
}
```


The property 'sparkTableName' is the name of the Spark view, visible when using the Spark Beeline tool (see below). Because the KPI data is stored by domain or by financial entity in Cassandra, a list of table names may be returned as the property 'cassandraTableNames'. The Cassandra table names can be queries in the Cassandra shell (see below).

In case the unique KPI code of the KPI is known, the following REST API can be used instead. It returns the same information as the previous API.

HTTP GET /api/analytics/kpiMetadata/<KPI code>/table-info

Starting with the March 2022 release of Adaptive UX, the following REST API returns the same information for all KPIs in a single request.

HTTP GET /api/analytics/kpiMetadata/table-info/all

Cassandra Shell

Cassandra provides a command-line shell *cqlsh* that can be used to execute CQL commands, the SQL-like language supported by Cassandra. The shell is especially useful for examining the contents of the browses or historical_kpi keyspaces, which contain all the Query Service browse data retrieved from the OpenEdge databases:

To run the Cassandra shell, Python 2.7 must be present on the command PATH.

Starting with the September 2021 release, a 'wrapper' script for cqlsh is provided by YAB that can be used to start it from the command line with no need to provide parameters such as port numbers and credentials. In this, change to the Adaptive UX home directory and run the following command:

```
scripts/cqlsh
```

For releases prior to September 2021, the native Cassandra script must be run with various command line parameters. To find the location of the shell utility and the Cassandra port number to which to connect, query the YAB configuration.

```
yab config cassandra.default.install.scripts.dir
```

```
yab config cassandra.default.node.main.native_transport_port
```

Change to the directory containing the Cassandra scripts. Start the shell, passing the required hostname and port number. The actual hostname (not 'localhost') must be used.

```
cd <Cassandra scripts directory>
./cqlsh <hostname> <Cassandra port>
```

Once the Cassandra shell has been started, determine the keyspace whose data you want to review and make it the default for all subsequent CQL commands.

```
cqlsh> use browses;
```

The shell can then be used to query the contents of the data lake with various commands, such as the following.

List the Browse Tables in the Browses Keyspace

The list shows the browse tables stored in the data lake. The list includes the domains or financials entities associated with the browse tables.

```
cqlsh:browses> describe tables;

kpi_357844472_12mex      kpi_1688405265_20fra     kpi__1355073026_10usa
kpi__846170097_12mex     kpi_1852174264_20fra     kpi__1863008167_22uk
kpi_1988561950_21nl      kpi_1988561950_12mex     kpi__692383936
kpi__1809347128_20fra    kpi__45865388_23ger      kpi__1451599441_10usa
kpi_1202448307_12mex     kpi_1044215493_21nl      kpi_1533800352_30chn
kpi_1202448307_30chn     kpi__1063222905_12mex    kpi_1384866991_10usa
kpi__186238821_12mex     kpi_1705405443_qad       kpi_1988561950_20fra
kpi_1650412229_11can     kpi__1798778746_20fra    kpi__1063222905_11can
kpi__1975819738_20fraco  kpi_1471838795_31aus     kpi_1105371684_10usa
kpi_1206509372_31aus     kpi_1384866991_11can     kpi__1423034166_21nl
kpi__1282164377_23ger    kpi__45865388_31aus      kpi__1809347128_12mex
kpi__1422684978_23ger    kpi__1319591205_22uk     kpi_642442275_10usaco
kpi__1942540096_12mex    kpi__1808255083_10usa    kpi__1604144893_31aus
kpi_1533800352_10usa     kpi__1423034166_11can    kpi__875669433
kpi__78780668_12mexco    kpi_1044215493_31aus     kpi_1529480452_31aus
kpi_1101437247_31aus     kpi_1175672099_12mex     kpi_1852174264_11can
kpi_1854608121_30chn     kpi_1471838795_21nl      kpi_1206509372_11can
kpi__1916630248_11can    kpi_1101437247_40brz     kpi_131336283_21nl
kpi_1080786763           kpi_131336283_22uk       kpi_220484148_12mex
kpi_514817284_10usa      kpi_1529480452_22uk      kpi__6799222_40brz
...

cqlsh:browses>
```

As of the September 2020 release of Adaptive UX, the KPI and browse cannot be identified based on the table name. The table names may be changed in future releases to facilitate easier inspection using the Cassandra shell.

Display Status Information About Each Table in the Browses Keyspace

This information comes from the Query Service table browsecopystatus. It contains the date-time when the results of each KPI browse were last copied into Cassandra.

```
cqlsh:browses> select * from browsecopystatus;

browsecopy                                    | amountcopied | canceled |
failed                          | finished                          |
started                         | submitted
-----------------------------------------------+--------------+----------
+---------------------------------+---------------------------------
+---------------------------------+---------------------------------
   e874314d-425a-59b6-6314-c157204f2414_31aus_ |            0 |     null
|                           null | 2020-08-23 00:49:06.683000+0000 | 2020-08-23 00:49:04.681000
+0000 | 2020-08-23 00:48:55.330000+0000
    988f1fb8-87a7-0cad-5714-ef2780ae4474_22uk_ |           64 |     null
|                           null | 2020-08-22 23:09:37.249000+0000 | 2020-08-22 23:09:35.410000
+0000 | 2020-08-22 23:09:27.684000+0000
   d1be0cc5-4c16-6c9b-6314-4e6830e1e6ed_20fra_ |            0 |     null
|                           null | 2020-08-23 00:46:22.606000+0000 | 2020-08-23 00:46:20.917000
+0000 | 2020-08-23 00:46:13.553000+0000
    ec8e0e0b-1866-2d80-5514-a431c803afa0_21nl_ |          747 |     null
|                           null | 2020-08-22 23:23:44.525000+0000 | 2020-08-22 23:23:40.774000
+0000 | 2020-08-22 23:23:40.127000+0000
   969a714f-4089-b4a1-5414-c84af01bc680_40brz_ |          819 |     null
|                           null | 2020-08-22 23:08:23.896000+0000 | 2020-08-22 23:08:16.922000
+0000 | 2020-08-22 23:08:09.055000+0000
   862404b7-980e-8885-5414-5663a00cee43_30chn_ |         4251 |     null
|                           null | 2020-08-22 23:03:03.404000+0000 | 2020-08-22 23:02:45.261000
+0000 | 2020-08-22 23:02:09.957000+0000
   cf4f03ad-e23f-5a99-6214-56f5481b6a0c_30chn_ |            0 |     null
|                           null | 2020-08-23 00:46:41.284000+0000 | 2020-08-23 00:46:39.181000
+0000 | 2020-08-23 00:46:32.473000+0000
   e3a23bcc-e989-7890-5814-7b9a40ce2e87_10usa_ |            1 |     null
|                           null | 2020-08-22 23:21:45.147000+0000 | 2020-08-22 23:21:44.142000
+0000 | 2020-08-22 23:21:43.601000+0000
   a84c4040-874b-9c98-5514-267450caeec0_23ger_ |          189 |     null
|                           null | 2020-08-22 23:12:18.824000+0000 | 2020-08-22 23:12:17.125000
+0000 | 2020-08-22 23:12:12.562000+0000
   ce6200b2-b081-75a5-6b14-c987c0b5eeb6_10usa_ |            0 |     null
|                           null | 2020-08-22 23:18:53.094000+0000 | 2020-08-22 23:18:51.696000
+0000 | 2020-08-22 23:18:50.636000+0000
   dbf98bf0-4d3e-b48d-5614-df4c980b1c9b_10usa_ |           65 |     null
|                           null | 2020-08-22 23:19:09.521000+0000 | 2020-08-22 23:19:08.502000
+0000 | 2020-08-22 23:19:07.997000+0000
   862404b7-980e-8885-5414-cfa1708c494f_40brz_ |         3435 |     null
|                           null | 2020-08-22 23:03:30.869000+0000 | 2020-08-22 23:03:20.566000
+0000 | 2020-08-22 23:02:10.463000+0000
   ec8e0e0b-1866-2d80-5514-b522e05c709e_12mex_ |            0 |     null
|                           null | 2020-08-22 23:24:28.509000+0000 | 2020-08-22 23:24:20.272000
+0000 | 2020-08-22 23:23:46.528000+0000
   ebce0a62-120c-bfa7-5714-023820b2c450_11can_ |         4120 |     null
|                           null | 2020-08-22 23:23:00.680000+0000 | 2020-08-22 23:22:45.467000
+0000 | 2020-08-22 23:22:27.670000+0000
   ba98dc66-9607-7698-5514-4e22e044cf41_30chn_ |         4452 |     null
|                           null | 2020-08-22 23:17:19.834000+0000 | 2020-08-22 23:17:06.574000
+0000 | 2020-08-22 23:15:40.269000+0000
 b7e49d22-1d54-81b5-5814-ea8a181d87df__10usaco |            0 |     null | 2020-08-26 17:33:
41.945000+0000 |                           null | 2020-08-26 17:33:28.697000+0000 | 2020-08-26
17:32:59.894000+0000
   e29f724f-acba-ed81-5914-c51ef00b477b_40brz_ |         2425 |     null
|                           null | 2020-08-22 23:21:11.908000+0000 | 2020-08-22 23:21:04.490000
+0000 | 2020-08-22 23:20:39.019000+0000
...

cqlsh:browses>
```

Starting with the September 2021 release, the browsecopystatus table was enhanced to include the Cassandra table name of each KPI browse. This information can be useful when the contents of a particular table must be inspected to determine the cause of some data refresh problem.

```
cqlsh:browses> select * from browsecopystatus;

 browsecopy                                    | amountcopied | canceled |
cassandratablename              | failed | finished                          |
```

```
started                          | submitted
-------------------------------------------------+--------------+----------
+---------------------------------+--------+--------------------------------
+---------------------------------+--------------------------------
   e874314d-425a-59b6-6314-c157204f2414_31aus_ |            0 |     null |    browses.
kpi_1688405265_31aus |   null | 2021-11-10 23:06:26.826000+0000 | 2021-11-10 23:06:21.481000+0000
| 2021-11-10 23:00:30.903000+0000
   9c616359-1492-15b8-7914-1a9c20cdd88c_23ger_ |         2894 |     null |    browses.
kpi__526967177_23ger |   null | 2021-11-10 23:08:29.545000+0000 | 2021-11-10 23:07:48.541000+0000
| 2021-11-10 23:02:02.200000+0000
    988f1fb8-87a7-0cad-5714-ef2780ae4474_22uk_ |           64 |     null |     browses.
kpi_1202448307_22uk |   null | 2021-11-10 23:37:28.697000+0000 | 2021-11-10 23:37:22.080000+0000
| 2021-11-10 23:27:12.686000+0000
   d6237112-8155-f9be-8414-e6bd207c691d_11can_ |            0 |     null |     browses.
kpi_369759295_11can |   null | 2021-11-10 23:15:21.786000+0000 | 2021-11-10 23:15:15.288000+0000
| 2021-11-10 23:10:47.241000+0000
   d1be0cc5-4c16-6c9b-6314-4e6830e1e6ed_20fra_ |            0 |     null |    browses.
kpi_1705405443_20fra |   null | 2021-11-11 21:15:26.091000+0000 | 2021-11-11 21:15:24.404000+0000
| 2021-11-11 21:15:16.601000+0000
 fe55c84d-d2eb-d895-8514-3f38d842b902__30chnco |           12 |     null | browses.
kpi__1338502100_30chnco |   null | 2021-11-10 22:41:48.772000+0000 | 2021-11-10 22:41:43.525000
+0000 | 2021-11-10 22:39:04.474000+0000
    ec8e0e0b-1866-2d80-5514-a431c803afa0_21nl_ |          747 |     null |     browses.
kpi__707214577_21nl |   null | 2021-11-10 22:48:08.856000+0000 | 2021-11-10 22:48:02.021000+0000
| 2021-11-10 22:41:23.859000+0000
   969a714f-4089-b4a1-5414-c84af01bc680_40brz_ |          882 |     null |    browses.
kpi_1650823036_40brz |   null | 2021-11-10 23:39:18.614000+0000 | 2021-11-10 23:39:05.535000+0000
| 2021-11-10 23:28:58.007000+0000
   862404b7-980e-8885-5414-5663a00cee43_30chn_ |         4578 |     null |    browses.
kpi_2080027167_30chn |   null | 2021-11-10 23:32:02.558000+0000 | 2021-11-10 23:31:32.717000+0000
| 2021-11-10 23:22:14.160000+0000
   e3a23bcc-e989-7890-5814-7b9a40ce2e87_10usa_ |            1 |     null |     browses.
kpi_887544142_10usa |   null | 2021-11-10 23:10:37.128000+0000 | 2021-11-10 23:10:22.459000+0000
| 2021-11-10 23:04:47.233000+0000
   a84c4040-874b-9c98-5514-267450caeec0_23ger_ |          189 |     null |   browses.
kpi__1355073026_23ger |   null | 2021-11-10 23:14:31.921000+0000 | 2021-11-10 23:14:26.290000
+0000 | 2021-11-10 23:10:39.826000+0000
   a450d92a-f743-6aa2-7914-8a8130a226be_40brz_ |            0 |     null |    browses.
kpi__540354320_40brz |   null | 2021-11-10 22:39:48.099000+0000 | 2021-11-10 22:39:45.343000+0000
| 2021-11-10 22:36:34.766000+0000
   dbf98bf0-4d3e-b48d-5614-df4c980b1c9b_10usa_ |           65 |     null |   browses.
kpi__1316390246_10usa |   null | 2021-11-10 23:20:52.576000+0000 | 2021-11-10 23:20:44.472000
+0000 | 2021-11-10 23:15:12.016000+0000
   862404b7-980e-8885-5414-cfa1708c494f_40brz_ |         3435 |     null |   browses.
kpi__1422684978_40brz |   null | 2021-11-10 23:30:17.400000+0000 | 2021-11-10 23:29:51.741000
+0000 | 2021-11-10 23:21:28.652000+0000
   db6b5995-686e-d18a-7514-efe2e0390e81_10usa_ |            0 |     null |   browses.
kpi__1403020869_10usa |   null | 2021-11-10 23:44:46.904000+0000 | 2021-11-10 23:44:37.727000
+0000 | 2021-
...

cqlsh:browses>
```

Remove the Browse Data

If the browse data in the Query Service ever became corrupted and needed to be entirely refreshed, no backup-restore procedure would be needed. Instead, the Cassandra data lake can be emptied and re-populated by processing the necessary browses against the current OpenEdge source databases.

Individual browse tables can be dropped from browses keyspace as follows.

```
cqlsh:browses> drop table <table name>;
```

Here is an example.

```
cqlsh:browses> drop table kpi__186238821_10usa;
```

Alternatively, the following command will drop all browse tables from the data lake. A timeout error may be displayed if processing requires more that the default 10-second limit. However, in this case the command will still be processed in the background despite the timeout.

```
cqlsh:browses> drop keyspace browses;
```

To re-populate the Cassandra keyspace with browse data, restart the Tomcat Web UI instance with cache warming enabled. All browses needed to support the KPIs used by the Action Centers are re-processed, and the results are copied into Cassandra.

To exit the shell, use the quit command.

```
cqlsh:browses> quit;
```

For more information about the shell and other Cassandra tools, see the documentation at the Apache Cassandra web site.

Spark Web User Interface

Spark provides a web user interface that displays historical information about the work it has performed for the Query Service. The display includes:

- A list of scheduler tasks and stages
- A summary of data set sizes and memory usage
- Environmental information
- Information about the running Spark executors

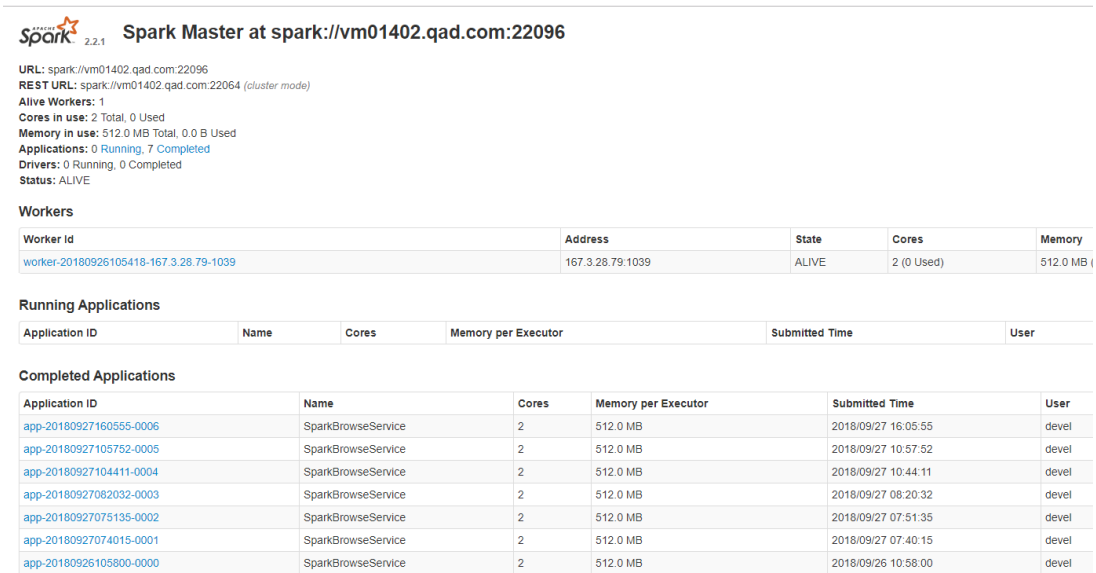
For more information, see the documentation at the Apache Spark web site.

To display the UI, query the YAB configuration to find the correct HTTP port to connect.

```
yab config spark.masterdefault.env.webui.port
```

In a web browser, go to the page http://<hostname>:<Spark Web UI port>

Below is a sample Spark display.



Spark 2.2.1 **Spark Master at spark://vm01402.qad.com:22096**

URL: spark://vm01402.qad.com:22096
REST URL: spark://vm01402.qad.com:22064 *(cluster mode)*
Alive Workers: 1
Cores in use: 2 Total, 0 Used
Memory in use: 512.0 MB Total, 0.0 B Used
Applications: 0 Running, 7 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20180926105418-167.3.28.79-1039	167.3.28.79:1039	ALIVE	2 (0 Used)	512.0 MB (

Running Applications

Application ID	Name	Cores	Memory per Executor	Submitted Time	User

Completed Applications

Application ID	Name	Cores	Memory per Executor	Submitted Time	User
app-20180927160555-0006	SparkBrowseService	2	512.0 MB	2018/09/27 16:05:55	devel
app-20180927105752-0005	SparkBrowseService	2	512.0 MB	2018/09/27 10:57:52	devel
app-20180927104411-0004	SparkBrowseService	2	512.0 MB	2018/09/27 10:44:11	devel
app-20180927082032-0003	SparkBrowseService	2	512.0 MB	2018/09/27 08:20:32	devel
app-20180927075135-0002	SparkBrowseService	2	512.0 MB	2018/09/27 07:51:35	devel
app-20180927074015-0001	SparkBrowseService	2	512.0 MB	2018/09/27 07:40:15	devel
app-20180926105800-0000	SparkBrowseService	2	512.0 MB	2018/09/26 10:58:00	devel

The page contains hyperlinks providing more details on particular tasks and links to native Spark log entries for them.

Spark Beeline Tool

While the Spark Web UI described above provides many operational details regarding Spark activities, queries, and environment characteristics, it does not provide a way to query the browse data that has been cached in the Query Service. However, Spark also includes beeline, a command-line utility that can connect to the Spark ThriftServer within the Query Service, and access the cached data using SQL.

To find the location of the Spark files, the ThriftServer port number to which to connect, and the required ThriftServer credentials, query the YAB configuration.

```
yab config packages.spark.dir
```

```
yab config spark-thriftserver.default.port
yab config spark-thriftserver.default.username
yab config spark-thriftserver.default.password
```

Change to the Spark package directory. Start the beeline tool, passing the required hostname and port number. The actual hostname (not 'localhost') must be used.

```
cd <Spark package directory>
./bin/beeline

>beeline !connect jdbc:hive2://<hostname>:<ThriftServer port>
Connecting to jdbc:hive2://<hostname>:<ThriftServer port>
Enter username for jdbc:hive2://vmlwebs20t:22178: <ThriftServer username>
Enter password for jdbc:hive2://vmlwebs20t:22178: <ThriftServer password>
log4j:WARN No appenders could be found for logger (org.apache.hive.jdbc.Utils).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Connected to: Spark SQL (version 2.4.5)
Driver: Hive JDBC (version 1.2.1.spark2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://<hostname>:<ThriftServer port>>
```

Alternatively, the above parameters can be passed in directly from the command line.

```
cd <Spark package directory>
./bin/beeline -u jdbc:hive2://<hostname>:<ThriftServer port> -n <ThriftServer username> -p
<ThriftServer password>

Connecting to jdbc:hive2://<hostname>:<ThriftServer port>
log4j:WARN No appenders could be found for logger (org.apache.hive.jdbc.Utils).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Connected to: Spark SQL (version 2.4.5)
Driver: Hive JDBC (version 1.2.1.spark2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 1.2.1.spark2 by Apache Hive
0: jdbc:hive2://<hostname>:<ThriftServer port>>
```

All Spark SQL commands must specify the name "qad_global_temp" of the Spark global temporary view, or no data will be shown. The global temporary view is essentially a virtual database that contains the datasets cached in the Query Service in the form of SQL views. Unfortunately, the global temporary view cannot be set as the default database for SQL commands, but must be specified explicitly in each one.

List the KPI Datasets in the Spark Global Temporary View

The list shows the datasets stored in the Query Service, most of which are populated directly from the tables in the Cassandra data lake.

```
0: jdbc:hive2://vmlwebs20t:22178> show tables from qad_global_temp;
+-----------------+-----------------------------------------------------+-------------+--+
|     database    |                      tableName                      | isTemporary  |
+-----------------+-----------------------------------------------------+-------------+--+
| qad_global_temp  | kpi_1006542657_10usa                                | true         |
| qad_global_temp  | kpi_1006542657_11can                                | true         |
| qad_global_temp  | kpi_1006542657_12mex                                | true         |
| qad_global_temp  | kpi_1006542657_20fra                                | true         |
| qad_global_temp  | kpi_1006542657_21nl                                 | true         |
```

```
| qad_global_temp  | kpi_1006542657_22uk                                | true        |
| qad_global_temp  | kpi_1006542657_23ger                               | true        |
| qad_global_temp  | kpi_1006542657_30chn                               | true        |
| qad_global_temp  | kpi_1006542657_31aus                               | true        |
| qad_global_temp  | kpi_1006542657_40brz                               | true        |
| qad_global_temp  | kpi_1006542657_80tst                               | true        |
| qad_global_temp  | kpi_1006542657_90trn                               | true        |
| qad_global_temp  | kpi_1006542657___74cfc248d7b221303172a6c252031069  | true        |
| qad_global_temp  | kpi_1006542657_qad                                 | true        |
...

335 rows selected (0.279 seconds)
0: jdbc:hive2://vmlwebs20t:22178>
```

Display the Contents of a Spark Dataset

The contents of any view (dataset) can also be displayed, although the data is often not very readable on a terminal console.

```
0: jdbc:hive2://vmlwebs20t:22178> select * from qad_global_temp.kpi_1006542657_10usa;
+-------------+--------------+------------------------+-----------------------
+-----------------------+-----------------------+----------------------+------------------------
+-------------------+-----------------------+-----------------------+-----------------------
+-----------------------+---------------------+--------------------------
+-----------------------+--------------------+---------------------------
+-------------------------------+-------------------------------+-------------------------------
+-------------------------------+-------------------------------+-------------------------------
+-------------------------------+-------------------------------+-------------------------------
+-------------------------------+-------------------------------+-------------------------------
+-------------------------------+-------------------------------+-------------------------------
+-------------------------------+-----------------------+---------------------+--------------------
+--------------------+------------------------+------------------------+---------------------
+-----------------+--+
| domaincode  | sequencenbr  |  in_mstr__in_avg_date  |  in_mstr__in_avg_iss  |
in_mstr__in_avg_sls  | in_mstr__in_cnt_date  | in_mstr__in_domain  |  in_mstr__in_iss_date  |
in_mstr__in_part  | in_mstr__in_proj_use  |  in_mstr__in_qty_all  | in_mstr__in_qty_nonet  |
in_mstr__in_qty_oh  | in_mstr__in_qty_ord  |  in_mstr__in_qty_req  |  in_mstr__in_rec_date  |
in_mstr__in_site  | local_variables__local_var02  | local_variables__local_var03  |
local_variables__local_var04  | local_variables__local_var06  | local_variables__local_var08  |
local_variables__local_var09  | local_variables__local_var10  | local_variables__local_var11  |
local_variables__local_var12  | local_variables__local_var34  | local_variables__local_var36  |
local_variables__local_var38  | local_variables__local_var52  | local_variables__local_var54  |
local_variables__local_var55  | local_variables__local_var57  | local_variables__local_var90  |
local_variables__local_var92  | local_variables__local_var94  | local_variables__local_var96  |
pt_mstr__pt_abc_qty  |  pt_mstr__pt_desc1  | pt_mstr__pt_desc2  | pt_mstr__pt_group  |
pt_mstr__pt_part_type  | pt_mstr__pt_prod_line  | pt_mstr__pt_status  | pt_mstr__pt_um  |
+-------------+--------------+------------------------+-----------------------
+-----------------------+-----------------------+----------------------+------------------------
+-------------------+-----------------------+-----------------------+-----------------------
+-----------------------+---------------------+--------------------------
+-----------------------+--------------------+---------------------------
+-------------------------------+-------------------------------+-------------------------------
+-------------------------------+-------------------------------+-------------------------------
+-------------------------------+-------------------------------+-------------------------------
+-------------------------------+-------------------------------+-------------------------------
+-------------------------------+-------------------------------+-------------------------------
+-------------------------------+-------------------------------+-------------------------------
+-------------------------------+-----------------------+---------------------+--------------------
+--------------------+------------------------+------------------------+---------------------
+-----------------+--+
| 10USA       | 1            | 2016-02-16 16:00:00.0  | 0.523489932900000000  |
0.523489932900000000  | NULL                  | 10USA              | 2017-04-24 17:00:00.0  |
01010             | 0                    | 8.000000000000000000  | 0E-18                  |
9.000000000000000000  | 0E-18                | 1183.000000000000000000  | 2017-04-19 17:00:00.0
| 10-100            | A                             |                          |
M                             | false                         | 223217.740000000000000000     |
0E-18                         | 0E-18                         | 0E-18                         |
0E-18                         | 0E-18                         | 0E-18                         |
0E-18                         | 0E-18                         | 0                             |
0E-18                         | 0                             | 0E-18                         |
Standard                      | 24801.970000000000000000      | 0E-18                         |
0E-18                | Medical Ultrasound  | fff                | ULTRA              |
DEVICES                | 10                    | ACTIVE             | EA              |
| 10USA       | 2            | 2017-10-14 17:00:00.0  | 0E-18              | 0E-
18                | NULL                  | 10USA              | 2017-10-15 17:00:00.0  |
```

```
01010            | 0                    | 0E-18                | 0E-18                  | 0E-
18                | 0E-18                | 0E-18                   | 2017-10-15 17:00:00.0  |
dsite-18         | A                             |                              |
M                           | false                        | 0E-18                          |
0E-18                       | 0E-18                        | 0E-18                          |
0E-18                       | 0E-18                        | 0E-18                          |
0E-18                       | 0E-18                        | 0                              |
0E-18                       | 0                            | 0E-18                          |
Standard                    | 0E-18                        | 0E-18                          |
0E-18               | Medical Ultrasound  | fff               | ULTRA              |
DEVICES               | 10                   | ACTIVE              | EA              |
| 10USA       | 3           | NULL                 | 0E-18                | 0E-
18                | NULL                 | 10USA               | NULL                   |
01010            | 0                    | 0E-18                | 0E-18                  | 0E-
18                | 0E-18                | 0E-18                   | NULL                   |
dsite-8          | A                             |                              |
M                           | false                        | 0E-18                          |
0E-18                       | 0E-18                        | 0E-18                          |
0E-18                       | 0E-18                        | 0E-18                          |
0E-18                       | 0E-18                        | 0                              |
0E-18                       | 0                            | 0E-18                          |
Standard                    | 0E-18                        | 0E-18                          |
0E-18               | Medical Ultrasound  | fff               | ULTRA              |
DEVICES               | 10                   | ACTIVE              | EA              |
| 10USA       | 4           | NULL                 | 0E-18                | 0E-
18                | NULL                 | 10USA               | NULL                   | 01010-
008         | 0                    | 0E-18                | 0E-18                  | 0E-
18                | 0E-18                | 0E-18                   | NULL                   |
10-100           | b                             |
|                             | false                        | 0E-18                          |
0E-18                       | 0E-18                        | 0E-18                          |
0E-18                       | 0E-18                        | 0E-18                          |
0E-18                       | 0E-18                        | 0                              |
0E-18                       | 0                            | 0E-18                          |
Standard                    | 0E-18                        | 0E-18                          |
0E-18               | Verify Link to CC   |                   |
|                        |                      | ACTIVE              | EA              |
| 10USA       | 5           | NULL                 | 0E-18                | 0E-
18                | NULL                 | 10USA               | NULL                   | 01010-
014         | 0                    | 0E-18                | 0E-18                  | 0E-
18                | 0E-18                | 0E-18                   | NULL                   |
10-100           | a                             |
|                             | false                        | 0E-18                          |
0E-18                       | 0E-18                        | 0E-18                          |
0E-18                       | 0E-18                        | 0E-18                          |
0E-18                       | 0E-18                        | 0                              |
0E-18                       | 0                            | 0E-18                          |
Standard                    | 0E-18                        | 0E-18                          |
0E-18               | this is a testret   |                   |
|                        |                      |                   | EA              |
...
+-------------+-------------+-----------------------+----------------------
+----------------------+----------------------+---------------------+------------------------
+-------------------+-----------------------+----------------------+------------------------
+----------------------+---------------------+-------------------------
+------------------------+-------------------+------------------------------
+--------------------------------+--------------------------------+--------------------------------
+--------------------------------+--------------------------------+--------------------------------
+--------------------------------+--------------------------------+--------------------------------
+--------------------------------+--------------------------------+--------------------------------
+--------------------------------+--------------------------------+--------------------------------
+--------------------------------+--------------------------------+--------------------------------
+--------------------------------+----------------------+---------------------+--------------------
+--------------------+------------------------+------------------------+--------------------
+-----------------+--+
745 rows selected (2.579 seconds)
0: jdbc:hive2://vmlwebs20t:22178>
```

To disconnect from the ThriftServer and exit beeline, use the '!quit' command.

```
0: jdbc:hive2://vmlwebs20t:22178> !quit
```

While inside beeline, use the '!help' command to get information about many other beeline commands.

```
0: jdbc:hive2://vmlwebs20t:22178> !help
```

Other Tools to Access Cassandra and Spark Data

In addition to the Cassandra and Spark command-line tools already described, it is possible to view the contents of Cassandra and/or Spark using third-party SQL client tools (examples: DBeaver, DbVisualizer, SQuirreL SQL) that support Cassandra or Spark JDBC drivers. The configuration details are specific to each tool and outside the scope of the current document, but for frequent users these tools may provide a more convenient, graphical, and usable alternative to the command-line utilities described here.

Enabling Debug Logging

In order to investigate the details of problems related to Action Centers and Query Service, it is sometimes necessary to enable debugging selectively for one or more of the components. Most of the common logging configuration changes to enable debugging can be done using YAB commands.

Any specialized logging configuration changes not supported by YAB properties and commands must be made manually in a text editor. If this is necessary, it should be done only in consultation with QAD Support personnel.

Tomcat

In order to investigate the details of problems related to Action Centers and Query Service, it is often necessary to enable debugging in the Tomcat instance supporting the Web UI. To accomplish this, the logging configuration file must be modified. To find the location of the Tomcat web app, query the YAB configuration.

```
yab config webapp.webshell.dir
```

The logging configuration file is named logback.xml, and is stored in the WEB-INF/config/ directory under the web app location. To obtain help about the relevant YAB properties and commands, run the following YAB command.

```
yab help logback.webshell
```

Debugging should be enabled selectively for relevant parts of the Tomcat environment, not globally for all Java classes. Depending on the problem to be debugged, the following Java packages and classes are usually the most important. These packages and classes are used as the 'NAMESPACE' entries referenced in the YAB help.

Component	Packages/Classes to Debug
Action Centers - Overall	com.qad.analytics.core, com.qad.analytics.core.mvc.controller.view, com.qad.analytics.core.mvc.controller.data, com.qad.analytics.core.composer, com.qad.analytics.core.lps, com.qad.analytics.core.service, com.qad.analytics.core.util
Query Service - Spark processing	com.qad.qracore.service.impl.spark
Query Service - Cassandra processing	com.qad.qracore.service.impl.BrowseDefinitionServiceImpl, com.qad.qracore.service.impl.CassandraCopyServiceContext, com.qad.qracore.service.impl.CassandraCopyServiceImpl, com.qad.qracore.service.impl.CassandraCopyTask, com.qad.qracore.service.impl.CassandraCopyTaskFactory

For example, to enable debug-level logging for the *com.qad.analytics.core.service* package, set the following YAB property.

```
logback.webshell.loglevel.com.qad.analytics.core.service=debug
```

Enabling debug-level logging can cause the Tomcat log file size to expand quickly. It should be used only selectively and for short periods of time in production environments.

After the desired YAB properties have been set, run the following YAB command to update the logging settings.

```
yab logback-webshell-update
```

Once the changes are saved, they take effect within a minute or so with no need to restart the Tomcat instance.

Logi Composer

When investigating problems in the display of visuals inside the Action Centers supported by Logi Composer, expanded panels, or the KPI screen, it is sometimes helpful to enable debugging for one or more of the Composer components or microservices. The 'zoomdata' component is generally the most useful one for debugging purposes. To do this, set the following YAB properties.

```
logi-composer.default.microservice.zoomdata.configuration.logging.level.com.zoomdata=debug
logi-composer.default.microservice.zoomdata-query-engine.configuration.logging.level.com.
zoomdata=debug
logi-composer.default.microservice.zoomdata-edc-postgresql.configuration.logging.level.com.
zoomdata=debug
```

Logi Platform Services

When investigating problems in the display of visuals inside the Action Centers supported by Logi Platform Services, expanded panels, or the KPI screen, it is sometimes helpful to enable debugging for the LogiPS Application Service and /or Data Service. To do this, set the following YAB properties.

```
logi-platform-services.default.service.application.loglevel=debug
logi-platform-services.default.service.data.loglevel=debug
```

After these properties have been set, restart Logi Platform Services. There is no need to restart Tomcat.

Logi Info

When investigating problems in the display of visuals or grids inside the Action Centers supported by Logi Info, expanded panels, or the KPI screen, it is sometimes helpful to enable debugging for the Logi plugin classes specific to Action Centers. Most of the common logging configuration changes to enable debugging can be done using YAB commands. To find the location of the Tomcat webapp, query the YAB configuration.

```
yab config webapp.analytics-logi.dir
```

The logging configuration file is named log4j.properties, and is stored in the WEB-INF/classes/ directory under the web app location. To obtain help about the relevant YAB properties and commands, run the following YAB command.

```
yab help log4j.analytics-logi
```

To enable debugging for the QAD Logi plugin classes, set the following YAB property to change the root log level from FATAL to INFO.

```
log4j.analytics-logi.loglevel=info
```

Do not set the Logi log level to DEBUG, as this causes many internal Logi log messages to be written that are unlikely to be helpful in Action Center problem diagnosis.

After the desired YAB properties have been set, run the following YAB command to update the logging settings.

```
yab log4j-analytics-logi-update
```

The Tomcat instance must be restarted for the new log level to take effect.

Action Center Disaster Recovery

Introduction

In a disaster recovery scenario where data stored in the Adaptive UX databases are lost or damaged and cannot be restored from current replicas for any reason, it is possible that some changes made to KPIs and Action Centers since the time of the most recent database backup will be lost, and must be manually re-entered once service is restored. This risk exists because of differences in the built-in capabilities of the various databases that support the Action Centers.

- **OpenEdge databases**: Contains the KPIs and browse definitions that define the data required for the visuals and Action Centers, as well as the dashboard resource identifiers and access permissions for each Action Center displayed on the Web UI menu.
- **Logi Composer PostgreSQL databases or Logi Platform Services H2 product database**: Contains the definition of all visuals, and the contents and layout of all Action Centers.
- **Cassandra data lake**: Contains current and historical snapshots of the KPI datasets extracted from the operational databases, mainly browse result sets.

This section summarizes the disaster recovery procedures required or recommended for Action Center data. However, it does not cover the specific YAB commands that implement database backups and restores. For this information, see the YAB Configuration and Administration Guide or use the `yab help` command.

OpenEdge vs Logi Databases

The OpenEdge databases are backed up regularly and frequently. In addition, they are protected in case of disaster by roll-forward logging and after-imaging, which allows transactions committed since the previous backup to be rolled forward into a backup copy of the database to bring it up to date. This is most important to protect the business and transaction data outside of analytics, but also applies to the KPI and Action Center definitions stored in the same qaddb database. Thus, all KPI and Action Center definitions added/modified/deleted since the disaster can be restored without data loss. However, these definitions may not be in sync with Logi Composer or Logi Platform Services, as described below.

The details of the Logi Composer PostgreSQL databases and older Logi Platform Services H2 database are very different, but the disaster recovery considerations and recovery approaches are very similar. In both cases, the Logi databases are backed up at the same time as the OpenEdge databases, and the backups will therefore be in sync. When the most recent backups for OpenEdge and Logi are restored following a disaster, their contents will agree. No in-flight transactions will be lost, as all commits are written to disk immediately, not buffered in memory and flushed to disk later. However, unlike the OpenEdge databases, write-ahead logging and roll-forward capability are not enabled for the Logi databases. While databases transactions since the last backup can be applied automatically to the restored OpenEdge databases to make them current as of the time of the service outage, the same cannot be done for the Logi databases. Th e Logi databases contain only metadata, such as visual and dashboard definitions, which are relatively static and not updated as frequently as the business data. Roll-forward capability is therefore not nearly as important as with other Adaptive UX data. However, this difference implies that following a disaster and roll-forward recovery, changes made to the Logi databases since the most recent backup may be recoverable only through manual re-entry. This section describes in more specific terms the information that could be lost.

Logi Composer Sync

Starting with the September 2022 release in environments where Logi Composer is being used, the following YAB command should be run during the recovery process following a service outage.

```
yab webapp-analytics-composer-api-sync
```

It is run automatically as part of the **YAB environment-restore** command.

This command ensures that there is a correctly identified Action Center and KPI in Composer for every Action Center, KPI, and visual in the qaddb database. For objects that do not exist in qaddb but exist in Composer, it deletes them from Composer. From objects that exist in qaddb but not in Composer, it adds them to Composer. Thus, it partially cleans up the Composer database to help bring it into agreement with AUX, although it cannot restore updates made to Action Centers and visuals in Composer since the most recent backup was taken. Such updates must be re-created manually with the help of the instructions in this section of the Implementation Guide.

New or Changed Visuals Published to the Gallery

The most common and frequent kind of change that affects Action Centers is the creation and modification of charts and other visuals. After a disaster, Action Center users should be notified that any visuals added or modified since the restored database backup must be re-created and re-published manually. If the backup was taken recently and the duration of the service outage was short, users may be able to recall and re-apply their changes without a large effort.

Often, visuals are created or modified in combination with changes to the KPI definition with which those visuals are associated. The last modified date-time of each KPI is stored in the OpenEdge database. Following roll-forward actions to

fully restore the OpenEdge database, it may be helpful to identify those KPIs that were modified since the restored backup using a database query. Users who created or use those KPIs in particular could then be requested to check the associated visuals for currency, as an additional reminder.

The following ABL query can be run in the Progress Editor of the NetUI to identify those KPIs modified since the backup:

```
for each KpiMetadata no-lock where LastModifiedDateTime > datetime-tz(<date-time of restored
backup>):
display KpiName LastModifiedUser LastModifiedDateTime.
```

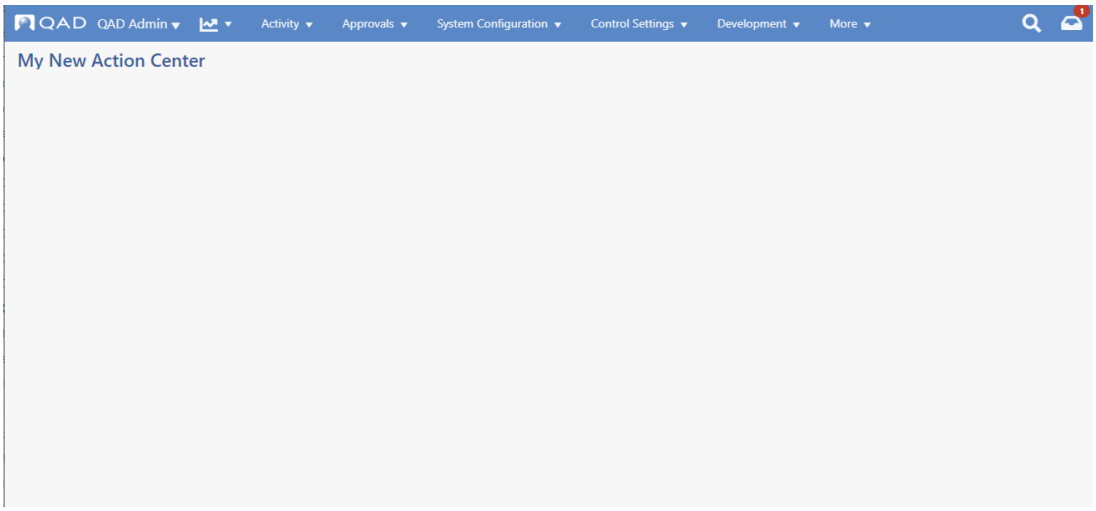
Alternatively, the following OpenEdge SQL query will also work in the Progress Editor:

```
select KpiName, LastModifiedUser, LastModifiedDateTime from KpiMetadata
where LastModifiedDateTime > datetime-tz(<date-time of restored backup>)
```

The date-time literal in the above expressions does not have to be quoted, but must be in the correct OpenEdge date-format for the database (for example, '05/31/2020' for US databases in 'MDY' format).

New Action Centers

If an Action Center was created since the last database backup prior to the service outage, following database restore and roll-forward actions it will be present on the Web UI menu, but the contents will not be present in Logi Composer or Logi Platform Services. In this case, displaying the Action Center from the Web UI will not raise an error, but simply display an empty dashboard.



The user can then re-create the Action Center using the same procedure as when it was created originally.

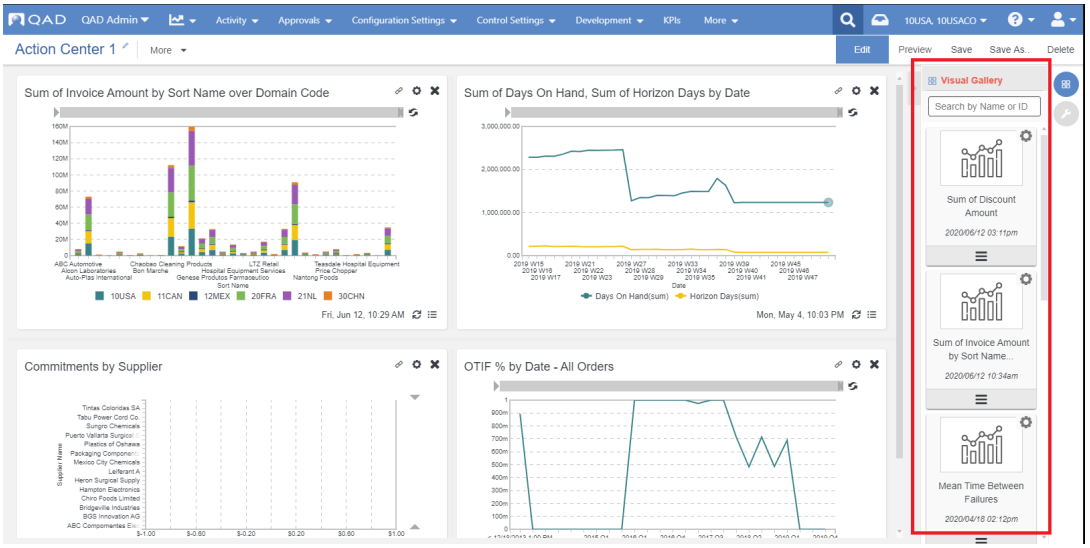
Modified Action Center Contents and Layouts

For an Action Center modified since the last database backup prior to the service outage, the recovery procedure is straightforward. The Action Center will display correctly from the Web UI menu with its old contents, and the user can re-apply the changes that were lost in the same way as the changes were made originally.

Deleted Visuals in Logi Platform Services

This section is relevant only to environments using Logi Platform Services, not Logi Composer.

Visuals that were deleted from Logi Platform Services since the last database backup prior to the service outage will have to be deleted again, once OpenEdge database restore and roll-forward actions are complete. To do this, a user with dashboard sharing permissions can display any Action Center that he/she is allowed to edit, click the Edit button, and press the Visual Gallery button  . The gallery will be displayed in a sidebar on the right side of the window.



Unwanted visuals can be selected with the help of the search bar at the top. Selected visuals are deleted from the gallery by pressing the cog icon and selecting the Delete command. However, any visual to be deleted must first be removed from Action Centers where it is being used, or the delete action will not be allowed.

Deleted Action Centers in Logi Platform Services

This section is relevant only to environments using Logi Platform Services, not Logi Composer.

Action Centers that were deleted from Logi Composer or Logi Platform Services since the last database backup prior to the service outage will be absent from the Web UI menu, once OpenEdge database restore and roll-forward actions are complete. Because they are no longer known and cannot be seen by the Web UI, they can be deleted only through Logi Platform Services directly. While this can be accomplished using internal Logi APIs, it is outside the scope of the present document to describe these details. It is recommended to contact QAD Service Delivery for assistance in this case (AB-26845).

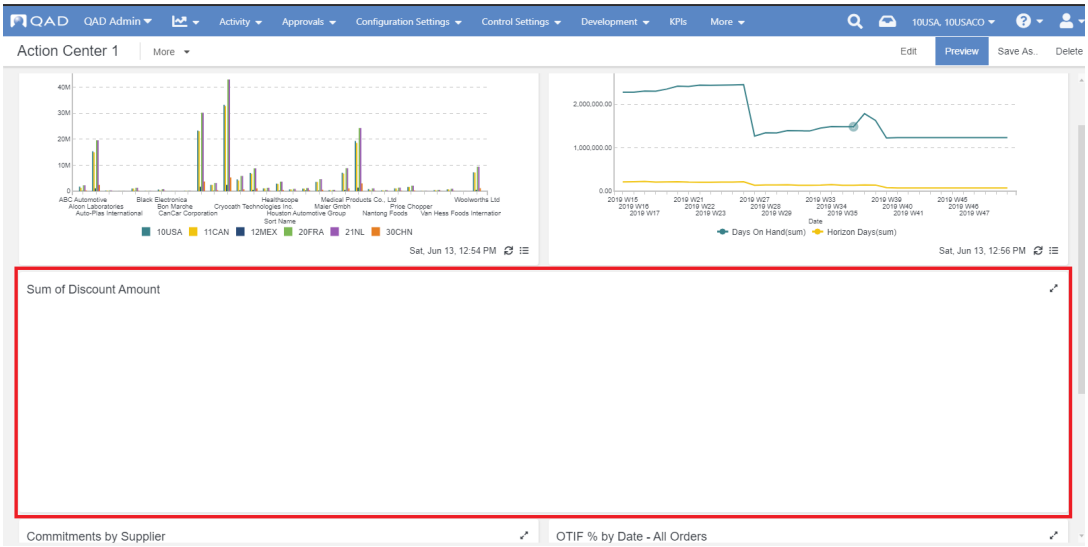
Because deleted Action Centers no longer exist in the Web UI at all, they cannot be accessed by any QAD user and, therefore, have no impact on other Adaptive UX functions.

What If Action Centers in Logi Composer or Logi Platform Services are Out of Sync with OpenEdge Data?

In most cases where lost Action Center contents cannot be re-created manually or will take some time to re-create, the out-of-date Action Centers and visuals should display without error with the old visuals and layouts. However, there are several exceptions where special attention is needed.

KPI Fields Removed or Renamed

One possibility is that a KPI field that is being used in visuals was deleted or renamed since the last restored backup. Standard Action Center validation prevents any field used in a visual from being deleted or renamed, and in this case the field would have had to be removed from all visuals before the delete or rename action. However, following disaster recovery the current OpenEdge database might be running with an older Logi Composer or Logi Platform Services database that still references the old field. In this kind of case, an Action Center panel containing the obsolete visual would display as blank.



In addition, the console log file for the tomcat-webui web server, normally `catalina.out`, would contain an error referencing the deleted/renamed field. In the following example, the 'Discount Amount' field that is used on the above visual 'Sum of Discount Amount' has been deleted.

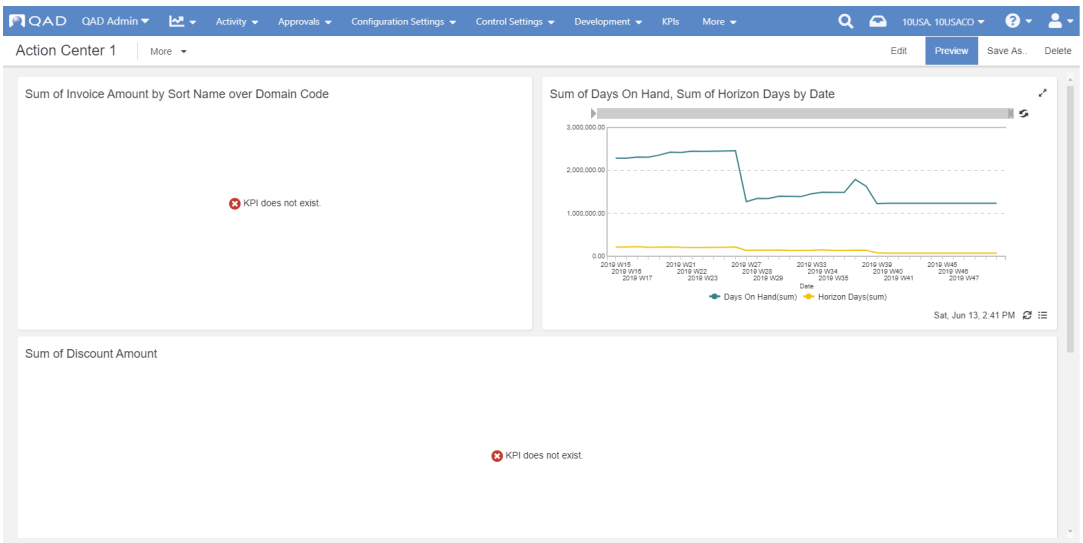
```
ERROR[HiveServer2-Background-Pool: Thread-1271] o.a.s.s.h.t.SparkExecuteStatementOperation:91
Error executing query, currentState RUNNING,
org.apache.spark.sql.AnalysisException: cannot resolve '`Query.Discount_Amount`' given input
columns: [Query.idh_hist__idh_qty_ord, Query.ih_hist__ih_ship, Query.ih_hist__ih_cust, Query.
idh_hist__idh_list_pr, Query.idh_hist__idh_qty_inv, Query.ih_hist__ih_bill, Query.
idh_hist__idh_due_date, Query.ad_mstr__ad_sort, Query.ih_hist__ih_nbr, Query.
local_variables__local_var03, Query.idh_hist__idh_part, Query.domainCode, Query.
idh_hist__idh_site, Query.pt_mstr__pt_desc1]; line 1 pos 59;
'GlobalLimit 201
+- 'LocalLimit 201
   +- 'Aggregate [count(1) AS Discount_Amount_count#14002L, 'COUNT('Query.Discount_Amount) AS
Discount_Amount_distinct#14003, 'MIN('Query.Discount_Amount) AS Discount_Amount_min#14004, 'MAX
('Query.Discount_Amount) AS Discount_Amount_max#14005, 'AVG(cast('Query.Discount_Amount as decimal
(28,5))) AS Discount_Amount_avg#14006, 'SUM(cast('Query.Discount_Amount as decimal(28,5))) AS
Discount_Amount_sum#14007]
      +- Filter upper(domainCode#8120) IN (10USA,10USA,11CAN,11CAN,12MEX,20FRA,21NL,21NL,22UK,
22UK,23GER,30CHN,31AUS,31AUS,40BRZ,80TST,90TRN,QAD)
         +- SubqueryAlias `Query`
            +- Project [ad_mstr__ad_sort#8122, to_utc_timestamp(idh_hist__idh_due_date#8124,
America/Los_Angeles) AS idh_hist__idh_due_date#14001, idh_hist__idh_list_pr#8126,
idh_hist__idh_part#8127, idh_hist__idh_qty_inv#8129, idh_hist__idh_qty_ord#8130,
idh_hist__idh_site#8131, ih_hist__ih_bill#8134, ih_hist__ih_cust#8137, ih_hist__ih_nbr#8141,
ih_hist__ih_ship#8143, local_variables__local_var03#8148, pt_mstr__pt_desc1#8150, domainCode#8120]
...
    at org.apache.spark.sql.catalyst.analysis.package$AnalysisErrorAt.failAnalysis(package.scala:
42)
    at org.apache.spark.sql.catalyst.analysis.
CheckAnalysis$$anonfun$checkAnalysis$1$$anonfun$apply$3.applyOrElse(CheckAnalysis.scala:111)
    at org.apache.spark.sql.catalyst.analysis.
CheckAnalysis$$anonfun$checkAnalysis$1$$anonfun$apply$3.applyOrElse(CheckAnalysis.scala:108)
    at org.apache.spark.sql.catalyst.trees.TreeNode$$anonfun$transformUp$1.apply(TreeNode.scala:
280)
    at org.apache.spark.sql.catalyst.trees.TreeNode$$anonfun$transformUp$1.apply(TreeNode.scala:
280)
    at org.apache.spark.sql.catalyst.trees.CurrentOrigin$.withOrigin(TreeNode.scala:69)
    at org.apache.spark.sql.catalyst.trees.TreeNode.transformUp(TreeNode.scala:279)
    at org.apache.spark.sql.catalyst.trees.TreeNode$$anonfun$3.apply(TreeNode.scala:277)
    at org.apache.spark.sql.catalyst.trees.TreeNode$$anonfun$3.apply(TreeNode.scala:277)
    at org.apache.spark.sql.catalyst.trees.TreeNode.
org$apache$spark$sql$catalyst$trees$TreeNode$$mapChild$2(TreeNode.scala:297)
    at org.apache.spark.sql.catalyst.trees.TreeNode$$anonfun$4$$anonfun$apply$13.apply(TreeNode.
scala:356)
    at scala.collection.TraversableLike$$anonfun$map$1.apply(TraversableLike.scala:234)
    at scala.collection.TraversableLike$$anonfun$map$1.apply(TraversableLike.scala:234)
    at scala.collection.mutable.ResizableArray$class.foreach(ResizableArray.scala:59)
    at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:48)
    at scala.collection.TraversableLike$class.map(TraversableLike.scala:234)
    at scala.collection.AbstractTraversable.map(Traversable.scala:104)
...
    at org.apache.spark.sql.SparkSession.sql(SparkSession.scala:642)
```

```
    at org.apache.spark.sql.SQLContext.sql(SQLContext.scala:694)
    at org.apache.spark.sql.hive.thriftserver.SparkExecuteStatementOperation.
org$apache$spark$sql$hive$thriftserver$SparkExecuteStatementOperation$$execute
(SparkExecuteStatementOperation.scala:232)
    at org.apache.spark.sql.hive.thriftserver.SparkExecuteStatementOperation$$anon$1$$anon$2.run
(SparkExecuteStatementOperation.scala:175)
    at org.apache.spark.sql.hive.thriftserver.SparkExecuteStatementOperation$$anon$1$$anon$2.run
(SparkExecuteStatementOperation.scala:171)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:422)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1893)
    at org.apache.spark.sql.hive.thriftserver.SparkExecuteStatementOperation$$anon$1.run
(SparkExecuteStatementOperation.scala:185)
    at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
    at java.util.concurrent.FutureTask.run(FutureTask.java:266)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    at java.lang.Thread.run(Thread.java:748)
```

In this case, the visual must be modified or re-created without referencing the KPI field that is no longer valid. The Action Center containing the visual must then be modified to include the replacement visual.

Browse Definition Modified

A similar problem occurs when the definition of the source browse associated with a KPI was modified since the last restored backup to remove fields that were being used by the KPI. In this case, following disaster recovery the current OpenEdge database and browse definitions might be running with an older Logi Composer or Logi Platform Services database with contents based on the old browse. In this kind of case, the KPI referencing the non-existent browse field would be broken.An Action Center panel containing visuals associated with that KPI would display with errors.



In addition, the console log file for the tomcat-webui web server would contain one or more errors referencing the KPI.

```
ERROR[http-bio-22011-exec-2] c.q.a.c.l.LogiPlatformWidgetSetupService:154
getLogiDashboardPanelSetupInfo(): error creating data visualization panel info for KPI code:
9be65599-692e-f782-6214-94e3c0caec26
java.lang.NullPointerException: null
    at com.qad.analytics.core.lps.DataLakeFieldService.lambda$2(DataLakeFieldService.java:60)
    at java.util.stream.ReferencePipeline$3$1.accept(ReferencePipeline.java:193)
    at java.util.stream.ReferencePipeline$2$1.accept(ReferencePipeline.java:175)
    at java.util.Iterator.forEachRemaining(Iterator.java:116)
    at java.util.Spliterators$IteratorSpliterator.forEachRemaining(Spliterators.java:1801)
    at java.util.stream.AbstractPipeline.copyInto(AbstractPipeline.java:482)
    at java.util.stream.AbstractPipeline.wrapAndCopyInto(AbstractPipeline.java:472)
    at java.util.stream.StreamSpliterators$WrappingSpliterator.forEachRemaining
(StreamSpliterators.java:312)
    at java.util.stream.Streams$ConcatSpliterator.forEachRemaining(Streams.java:742)
    at java.util.stream.AbstractPipeline.copyInto(AbstractPipeline.java:482)
    at java.util.stream.AbstractPipeline.wrapAndCopyInto(AbstractPipeline.java:472)
    at java.util.stream.ReduceOps$ReduceOp.evaluateSequential(ReduceOps.java:708)
    at java.util.stream.AbstractPipeline.evaluate(AbstractPipeline.java:234)
    at java.util.stream.ReferencePipeline.collect(ReferencePipeline.java:499)
```

```
    at com.qad.analytics.core.lps.LogiPlatformWidgetSetupInfoService.fillDataLakeInfo
(LogiPlatformWidgetSetupInfoService.java:195)
    at com.qad.analytics.core.lps.LogiPlatformWidgetSetupInfoService.
setUpLogiKpiBasedWidgetSetupInfo(LogiPlatformWidgetSetupInfoService.java:178)
    at com.qad.analytics.core.lps.LogiPlatformWidgetSetupInfoService.
getLogiDashboardPanelSetupInfo(LogiPlatformWidgetSetupInfoService.java:148)
    at com.qad.analytics.core.lps.LogiPlatformWidgetSetupService.getLogiDashboardPanelSetupInfo
(LogiPlatformWidgetSetupService.java:152)
...
    at com.qad.analytics.core.lps.LogiPlatformWidgetSetupService.getLogiDashboardPanelSetupInfos
(LogiPlatformWidgetSetupService.java:142)
    at com.qad.analytics.core.lps.mvc.controller.data.WidgetSetupController.
setupDashboardPanels_aroundBody14(WidgetSetupController.java:98)
    at com.qad.analytics.core.lps.mvc.controller.data.WidgetSetupController$AjcClosure15.run
(WidgetSetupController.java:1)
    at org.aspectj.runtime.reflect.JoinPointImpl.proceed(JoinPointImpl.java:149)
    at com.qad.qracore.mvc.interceptor.BEExtensionInterceptorImpl.executionAround
(BEExtensionInterceptorImpl.java:143)
    at com.qad.webshell.aspects.BEExtensionAspect.executionAround(BEExtensionAspect.java:28)
    at com.qad.analytics.core.lps.mvc.controller.data.WidgetSetupController.
setupDashboardPanels_aroundBody16(WidgetSetupController.java:96)
    at com.qad.analytics.core.lps.mvc.controller.data.WidgetSetupController$AjcClosure17.run
(WidgetSetupController.java:1)
...
```

In this case, multiple steps are generally required to recover:

1. Any visuals associated with the KPI that reference the deleted field must be either deleted or modified to remove the obsolete field reference.
2. The KPI definition must be re-configured in the KPI screen for the updated browse by pressing the Configure button to display the browse, then pressing OK. Once the KPI definition is correct, the KPI can be saved.
3. If necessary, existing visuals must be modified or new ones created for the KPI that reference the correct data.
4. Any Action Center containing visuals that were removed in an earlier step should be modified to include replacement visuals.

Cassandra Data Lake

The Cassandra database is used for several purposes related to Action Centers. It contains multiple keyspaces that must be considered separately.

browses Keyspace

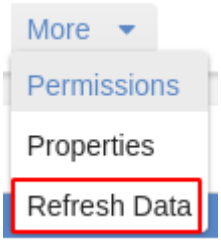
In the Cassandra data lake, the **browses** keyspace is used to hold current KPI data as of the latest refresh, whether scheduled or manual. The datasets may be accessed on line from Action Centers displayed in the Web UI.

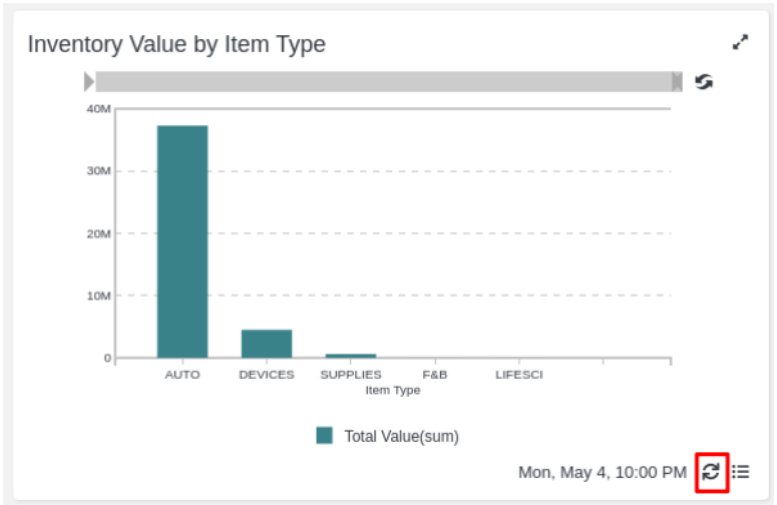
The detailed KPI and browse data stored in the browses keyspace are normally not backed up, as they are created from the current contents of the OpenEdge databases. In case of disaster, the contents of this Cassandra keyspace can be re-created using the standard scheduled refresh and cache warming processes, once those databases have been recovered following the service outage. Hence, there is virtually no risk that the KPI datasets directly consumed by and displayed in Action Center visuals would be permanently lost.

In case of disaster, the OpenEdge databases should first be restored using normal procedures. Once this is done and Cassandra is running again, the browses keyspace will be re-populated automatically at tomcat-webui startup, assuming the cache warming is enabled, and Action Center will display normally with current data from the OpenEdge databases.

If cache warming is not enabled, the KPI data must be re-created using one of the following options:

- **Scheduled refresh**: For those KPIs that are configured to refresh daily, weekly, or monthly, their Cassandra data will be populated automatically at the next scheduled refresh. In this case, however, Action Center users would have to wait to see current data for these KPIs.
- **Manual refresh**: Web UI users of Action Centers who are authorized to manually refresh KPIs that have been configured with manual refresh enabled can do so from the Action Center screens. Under the More menu on the Action Center toolbar, there is a Refresh Data command that will re-populate all Cassandra tables for the KPIs used in that Action Center. Data refreshes for particular dashboard panels can also be triggered by pressing the refresh icon in the lower right-hand corner of each panel.





historical_kpi Keyspace

The **historical_kpi** keyspace is used to hold KPI data from historical snapshots taken at scheduled date-times, which may extend years into the past. Unlike the browses keyspace, its contents cannot be re-created from the current OpenEdge databases. In this case, Cassandra is the system of record for the data and should be backed up at the same time the OpenEdge databases are backed up each day. This can be accomplished by ensuring that the historical_kpi keyspace is absent from the blacklist of Cassandra keyspaces excluded from the YAB environment-backup operation, listed in the property cassandra.default.node.backup.blacklist. YAB backs up the designated keyspaces using the Cassandra 'nodetool snapshot' command.

If there is a service outage while data is being written into Cassandra, no in-flight transactions are lost. Cassandra first writes all changes to a commit log before treating them as successful. If the system crashes before those writes have been saved in the persistent database tables, the commit log is automatically replayed when the node is restarted to read them into in-memory 'memtables', from which they are flushed to disk.

Once the Cassandra environment has been restored following a disaster, the historical_kpi keyspace will contain all data as of the time of the service outage. However, historical snapshots that were in process and interrupted at the time of the service outage may contain only a portion of the required data, and therefore may not be usable. In this case, the snapshot failure will be detected at tomcat-webui startup, and a new historical snapshot will be taken automatically to replace each one that failed. If new activity has been processed by Adaptive UX since the restoration of service but before replacement snapshots are created, it is possible that the contents of the new snapshots will be different than the contents that would have been included in the respective failed snapshots. However, in most disaster recovery scenarios this window of time will be short, reducing the risk that the new KPI snapshot will be inaccurate by a significant amount.

Logi Composer Migration Guide

Table of Contents

Migration Overview

The purpose of this guide is to walk through the steps required to migrate Action Centers, visuals, and dataviews (KPIs) from the Logi Platform Services tool used in previous Adaptive UX releases to Logi Composer. Starting with the September 2022 release of AUX, Logi Composer is enabled by default and Logi Platform Services is disabled. It is possible to avoid or postpone the use of Logi Composer and to retain Logi Platform Services in a September 2022 environment, but QAD strongly recommends that Composer be used instead of Logi Platform Services. Composer has richer functional capabilities, renders Action Centers in a more engaging UI, and is more usable generally. It also has better run-time performance.

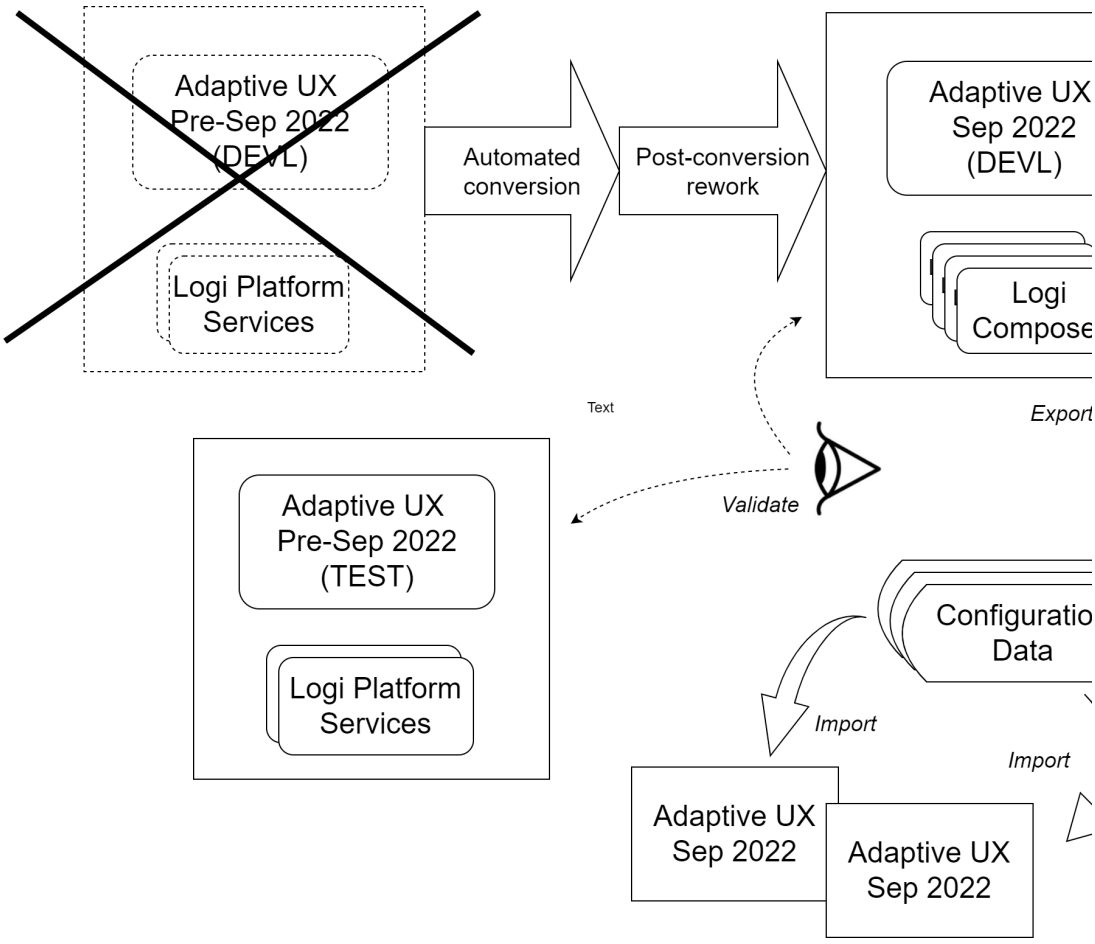
> Logi Platform Services is deprecated as of the September 2022 release, and will be obsoleted entirely in future AUX releases.

Starting with the September 2022 AUX release, all pre-defined QAD Action Centers and visuals are provided in Composer-compatible form only. The previous versions of these Action Centers for Logi Platform Services will no longer be maintained or included in any AUX release.

In any one AUX environment, either Logi Composer or Logi Platform Services may be used, but not both. During the update process, when the objects are being migrated, both Composer and Logi Platform Services are running. However, once the system administrator signals that the migration has been completed, Logi Platform Services is disabled and only Composer is used. The migration is one way only; there is no reverse migration from Composer back to Logi Platform Services. The migrated AUX environment is connected only to Composer, and no longer to Logi Platform Services. Thus, for each AUX environment, the decision to move to Composer is an irreversible one.

Hybrid Automated-Manual Migration Process

The migration to Composer is a mix of automated and manual steps. When the environment is updated with Composer enabled, all Action Centers, visuals, and related objects in the environment that were not provided by QAD are converted from Logi Platform Services to Composer automatically. This automated step performs the great majority of the migration work, avoiding the need for Action Centers users to recreate their Action Centers and visuals from scratch in the Composer tool. However, some calculations, filters, and visuals cannot be converted automatically to Composer, for various reasons. Most of this guide covers the details of how to review the converted objects, find gaps/errors, and fix them through rework and/or partial re-runs of the migration process.



Once the Action Centers and related objects have been converted to Composer, they must be reviewed for correctness. To facilitate this review, QAD strongly recommends that one AUX environment be upgraded to September 2022 and migrated to Composer first, while another environment, that has not yet been upgraded, remains available for use. Ideally, the two environments should contain almost the same business data. For most customers who maintain a set of controlled development, test, and production environments; the development environment would generally be upgraded to September 2022 and migrated to Composer first, while the test environment remains available for validation purposes. This approach makes before versus after comparisons relatively easy, as the configuration, along with the content of each visual in the upgraded Composer environment, can be checked by users against the same visuals in the Logi Platform Services environment.

Deferring the Migration

As mentioned above, migration from Logi Platform Services to Logi Composer is initiated automatically, by default, when an existing AUX environment is updated to the September 2022 release.

To upgrade an environment to September 2022, without enabling Logi Composer, add the following YAB property to `configuration.properties` before running the YAB update:

```
qad-analytics-core.composer.enabled=false
```

The updated environment will continue to use Logi Platform Services, without migrating any Action Centers or visuals. When the decision is made later to migrate the environment from Logi Platform Services to Logi Composer, set the above property to `true` and run another YAB update. The automated migration will run at the time Composer is enabled, unless the `qad-analytics-core.lps.composer-migration-complete` property has been set to `true` (see below).

Disabling Automated Migration

For environments upgrading to AUX September 2022 that contain only a few non-QAD Action Centers to be migrated to Composer, it may be simpler to disable the automated migration steps entirely. Some possible reasons for this approach are as follows:

- Using the automated migration process completes much of the conversion automatically, but still requires users to diagnose and fix errors for those portions of the Action Centers and visuals that could not be converted automatically. Re-implementing the Action Centers from scratch in Composer requires more manual work, but this work is less technical in nature and can be done entirely within the Web UI.
- If the number of Action Centers is small, it may be less work for Action Center owners to rebuild the Action Center contents from scratch using the powerful Composer UI features, with no need to review log files and to run a migration tool, in addition to performing some manual rework.
- Logi Composer is much more powerful than Logi Platform Services, and the products of an automated migration will not use all the new Composer features that end users may want in their new Action Centers. Often, the usability and value of an Action Center can be significantly improved by rebuilding it from scratch using Composer features and visual types that were not available in Logi Platform Services.

For environments containing many non-QAD Action Centers and visuals to be migrated, using the automated migration approach is recommended, as it usually requires significantly less manual work.

To upgrade an environment to AUX September 2022 with automated migration disabled, complete the following steps:

1. Add the following YAB property to `configuration.properties` before running the YAB update:

```
qad-analytics-core.lps.composer-migration-complete=true
```

2. Run the YAB update:

```
yab update
```

3. After the YAB update has completed successfully, run the following YAB command. It moves all Action Centers from their origin apps to Configuration Data, and modifies various identifiers in Web UI and Composer to build valid links between them.

```
yab webapp-analytics-composer-api-sync
```

Following these steps, the Composer-based Action Centers provided by QAD will be present, but all other Action Centers, developed in previous AUX releases using Logi Platform Services, will be gone. They will need to be re-implemented from scratch using the Composer functionality embedded in the Web UI of the September 2022 release.

Managing Common Action Centers After Migration

For organizations who have created and use a common set of Action Centers across multiple environments, there is no reason to complete the automated and manual Composer migration steps multiple times on the same objects. Instead, all the migration steps should be completed in one environment and exported as Configuration Data. After other environments have been upgraded to the September 2022 release with Composer enabled, the exported archives containing the migrated objects can be imported directly as Configuration Data, with no need for further migration effort. As of the September 2022 AUX release, all Action Centers are exchanged and distributed across AUX environments through the Web UI Configuration Data screen, rather than by installing apps. This change makes Action Centers easier to share in most environments, without the system administration overhead of app generation and installation.

For AUX environments being updated to the September 2022 release with Composer enabled, but for which no conversion of objects is necessary, complete the same steps described in the above section, Disabling Automated Migration.

After the environments have been upgraded to the September 2022 release, the converted Action Centers, previously exported from another upgraded environment, can be imported using the Configuration Data screen.

Prepare for Migration

Before migrating the non-QAD Action Centers and visuals to Logi Composer, as part of the upgrade to the AUX September 2022 release, several critical steps must be completed.

1. Make Sure Composer is Enabled

Logi Composer is enabled by default in the AUX September 2022 release, so in most cases, no action is required here. However, if the `qad-analytics-core.composer.enabled` YAB property has been configured manually, make sure that it is set to `true` in order to migrate Action Centers and related objects to Composer.

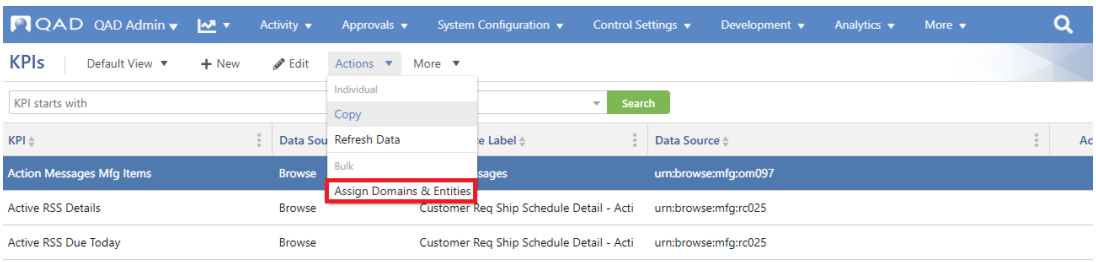
```
$ yab config qad-analytics-core.composer.enabled
qad-analytics-core.composer.enabled=true
```

If this property is set to `true`, the migration process runs automatically one time to convert all non-QAD Action Centers from Logi Platform Services to Logi Composer, and the Web UI will access only Composer to display Action Centers and visuals. If the property is not set to `true`, nothing will be migrated and the Web UI will continue using Logi Platform Services. Also, the latest QAD-provided Action Centers will not be available, as they are supported using Composer only.

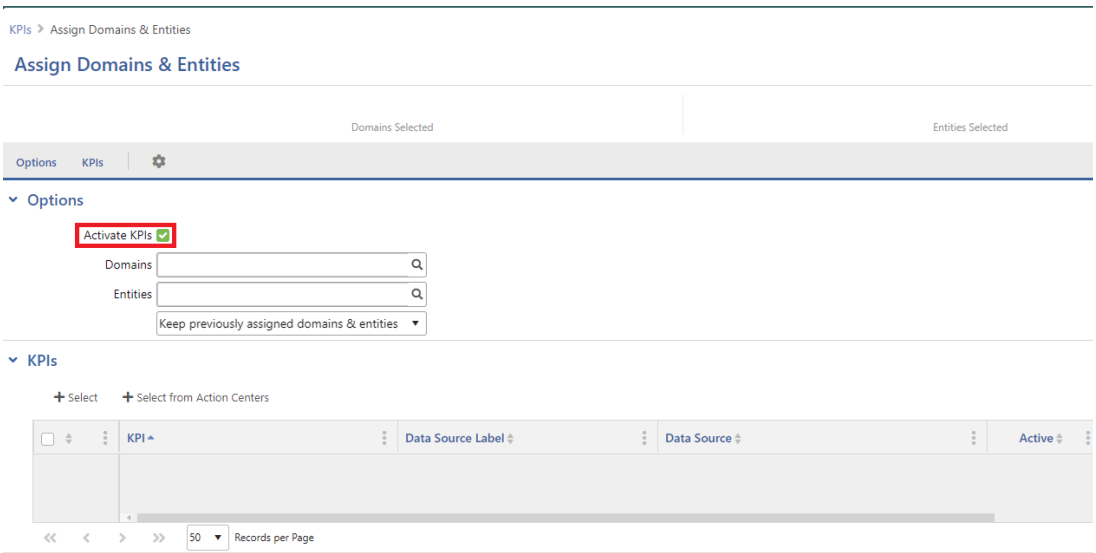
2. Set KPIs to the Active Status

The migration process works for active KPIs only. KPIs that are not set to the Active status will be skipped and will not be available in Composer, along with all their visuals and any Action Centers that use them. Therefore, before updating to AUX September 2022, make sure that all KPIs that are needed in Composer have been set to the Active status.

One or more KPIs can be activated in a single action using the Assign Domains & Entities bulk action on the KPIs screen.



From the pop-up window, you can select and activate any KPIs in the system.



The KPI activation takes place in the background, and if you select many KPIs, it can take time. Display the Web UI Background Processing screen to check the status of the background job, and make sure that it has completed successfully before starting the next step of this procedure.

3. Run Action Center Repair

Before updating to AUX September 2022 and migrating to Logi Composer, ensure that the KPIs and corresponding dataviews in Logi Platform Services are correct and in sync. In particular, the query used by Logi Platform Services to retrieve the data for a KPI must be correct, or else the migration of that KPIs dataview to Composer is likely to fail. Errors can sometimes be introduced into this query, if fields are added to the KPI and its associated browse, some time after the KPI was originally created. While this is not likely, QAD strongly recommends that the `action-center-dashboard-repair` command is run using YAB before upgrading to AUX September 2022, to synchronize the KPIs and their associated Logi Platform Services dataviews.

```
$ yab action-center-dashboard-repair
```

Make sure that the Action Center repair has completed successfully before starting the next step of this procedure.

4. Ensure Old Action Centers and Visuals Are Available in a Separate Environment

Once the environment has been updated and migrated to Logi Composer, it is critical to compare the migrated Action Centers and visuals, against their counterparts in Logi Platform Services, to verify the correctness of the migrated objects and to facilitate any rework. Therefore, a separate AUX environment, not yet upgraded to the September 2022 release, that contains the original Action Centers, visuals, and KPIs should be available during the migration process. This environment will be used for comparison purposes. Ideally, it should contain the same, or nearly the same, business data as the environment being migrated, so that the contents of each Action Center and visual can be readily compared.

5. Back Up the Source Environment

Before upgrading to AUX September 2022 and migrating to Composer, back up the source environment. While backups would likely be performed routinely before any upgrade, we recommend taking a backup just after the preceding steps have been completed. Then, the preceding steps will not have to be completed again if the backup needs to be restored and the current procedure restarted for any reason.

Run Automated Migration

This page describes the recommended steps for running the automated portion of the Logi Composer migration, which is, by default, a standard step in the AUX September 2022 release upgrade.

1. Migrate from Logi Platform Services to Logi Composer

Migrating the non-QAD Action Centers and related objects to Logi Composer can be done at the same time as the AUX September 2022 upgrade (the default approach) or later.

Run During the Update to AUX September 2022

By default, upgrading any AUX environment that is using Logi Platform Services to the September 2022 release will automatically cause all non-QAD Action Centers, dataviews (KPIs), and visuals to be converted and migrated to Composer. This step is run as part of the YAB update, unless the `qad-analytics-core.composer.enabled` YAB property is set to `false`.

Run After the Update to AUX September 2022

The migration to Composer may be deferred until sometime after the AUX September 2022 upgrade by setting the `qad-analytics-core.composer.enabled` YAB property to `false`, as mentioned above. In this case, the migration is performed during the next YAB update after, the property has been reset to `true`.

```
qad-analytics-core.composer.enabled=true
```

Run Separately from the YAB Update

The migration to Composer is performed by the `yab update` command only once. Once migration has been run once, further YAB update executions will not trigger Composer migration. However, the migration can be re-run outside of YAB update using the `logi-composer-default-lps-migrate-mode-append` YAB command.

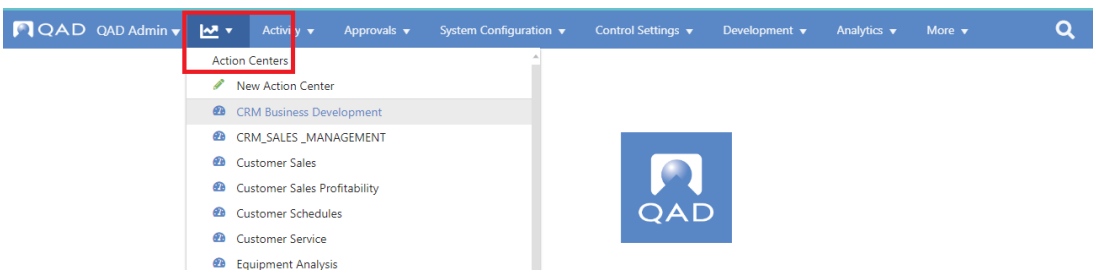
```
yab logi-composer-default-lps-migrate-mode-append
```

Reruns may be required during the migration process in order to fix migration errors. These scenarios are covered in more detail in later sections of this document.

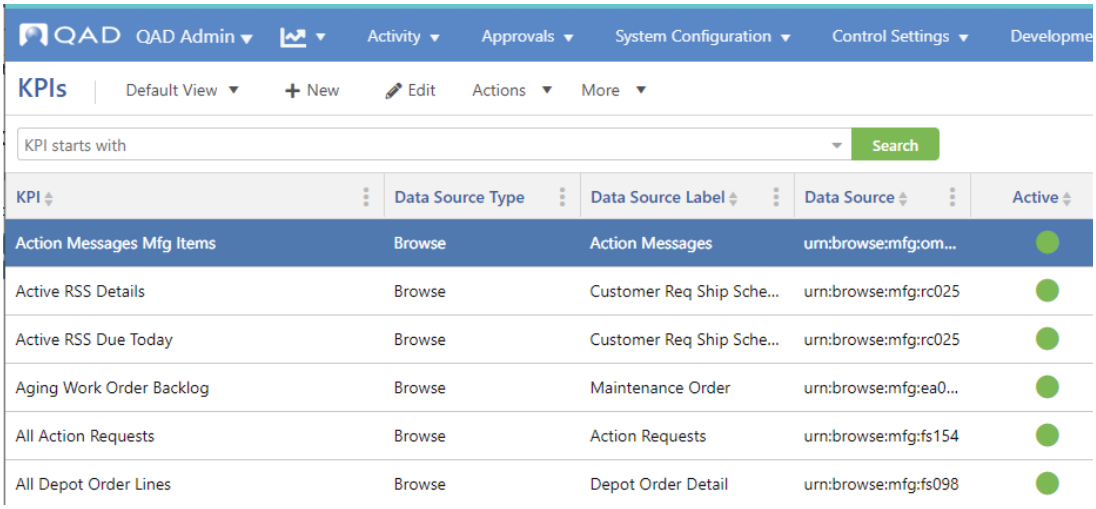
2. Check the Migrated Action Centers and Note Gaps

Once the migration process has been run using YAB, multiple review steps are required. The first step is to check whether all expected non-QAD Action Centers and KPIs were migrated to Composer from Logi Platform Services.

To check this, compare the Action Centers listed in the Web UI menu against the expected list from before the upgrade. Note any missing entries.



Next, check the KPIs listed in the KPI screen against the expected list from before the upgrade. Note any missing entries.

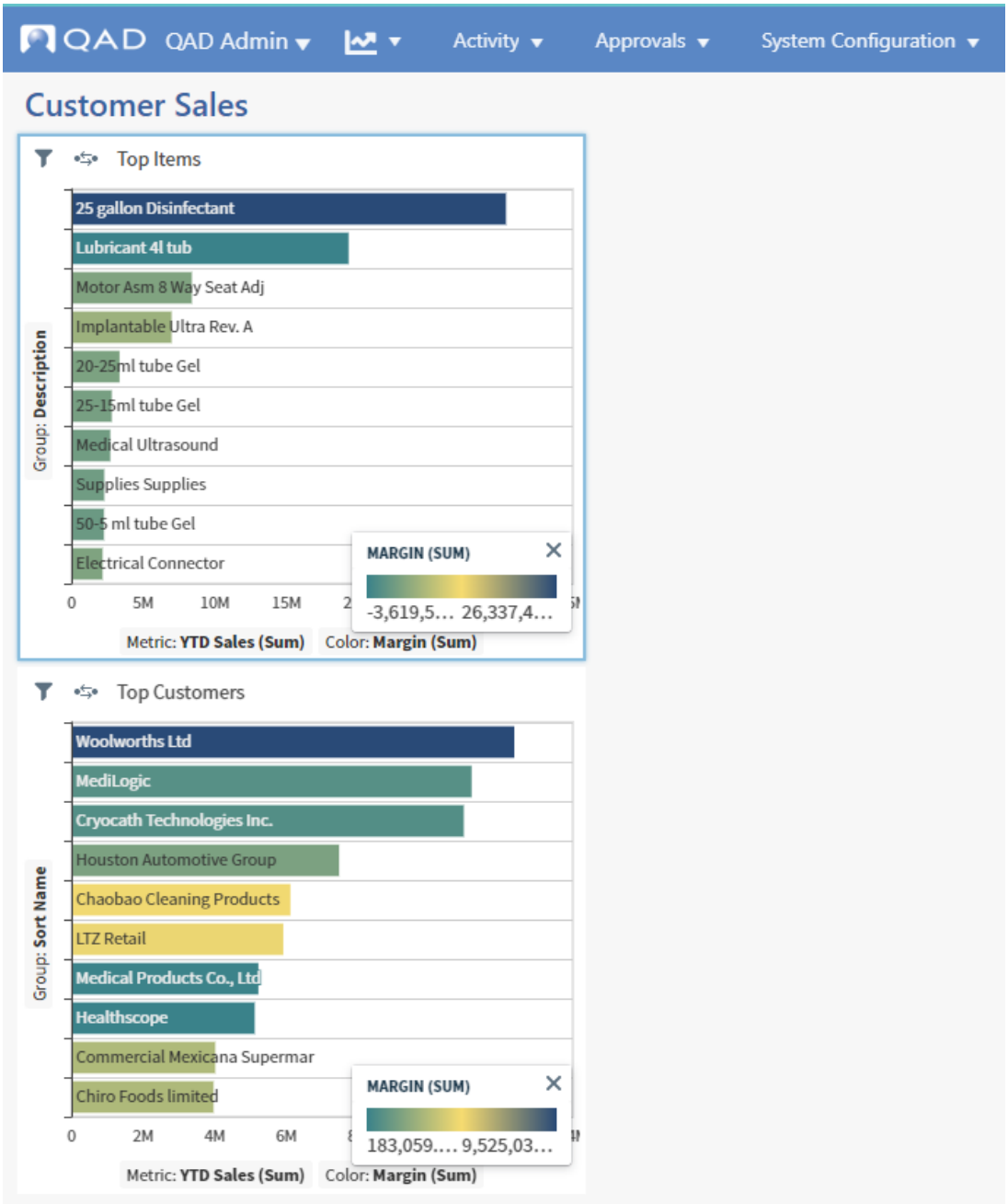


The omissions will be investigated in the Review and Repair Migration Gaps step of this procedure.

3. Review the Action Center Content and Note Gaps

Next, display each of the migrated Action Centers in the Web UI and note any missing panels. Some panels and visuals, or their associated dataview (KPI), may have been skipped by the migration because of unsupported syntax. The specific kinds of errors that may be encountered will be covered in the Review and Repair Migration Gaps step of this procedure.

Also, you will see that all panels are aligned under a single column, rather than in multiple rows and columns across the screen. The layout will be corrected in the Clean Up Working Action Centers and Visuals step of this procedure.


QAD
QAD Admin
Activity
Approvals
System Configuration
Customer Sales
Top Items
25 gallon Disinfectant
Lubricant 4l tub
Motor Asm 8 Way Seat Adj
Implantable Ultra Rev. A
20-25ml tube Gel
25-15ml tube Gel
Medical Ultrasound
Supplies Supplies
50-5 ml tube Gel
Electrical Connector
Group: Description
0
5M
10M
15M
MARGIN (SUM)
-3,619,5… 26,337,4…
Metric: YTD Sales (Sum)
Color: Margin (Sum)
Top Customers
Woolworths Ltd
MediLogic
Cryocath Technologies Inc.
Houston Automotive Group
Chaobao Cleaning Products
LTZ Retail
Medical Products Co., Ltd
Healthscope
Commercial Mexicana Supermar
Chiro Foods limited
Group: Sort Name
0
2M
4M
6M
MARGIN (SUM)
183,059.… 9,525,03…
Metric: YTD Sales (Sum)
Color: Margin (Sum)

Review and Repair Migration Gaps

In the previous section, missing Action Centers and visuals were identified by reviewing the migrated Action Centers in the Web UI. This section describes various cases where dashboards (Action Centers), visuals, or dataviews (KPIs) may have been skipped during the migration process. The specific causes and corrective actions are different for each case.

Review Errors in the appendResult File

Once any missing Action Centers or Action Center visuals have been noted, open and review the errors identified in the `appendResult` file created by the migration process. The name of the `appendResult` file is referenced in the `yab.log` file, near the end of the log entries for the `logi-composer-default-lps-migrate-mode-append` process.

```
...
2022-09-12 13:47:20,812 DEBUG [Thread-3:] STDOUT - [2022-09-12 20:47:20.810][info] - Result file
'appendResult1663015640792.json' was created successfully.;
2022-09-12 13:47:20,869 DEBUG [Thread-3:] STDOUT - [2022-09-12 20:47:20.869][info] - Appended
finished!;
2022-09-12 13:47:20,869 DEBUG [Thread-3:] STDOUT - [2022-09-12 20:47:20.869][info] - Process was
done!;
2022-09-12 13:47:20,940 DEBUG [main:92bb] APPLY - logi-composer-default-lps-migrate-mode-append
UPDATED
...
```

The location of this file is in the migration tool directory, which is stored in the `logi-composer.default.migration-tool.dir` YAB property.

```
$ yab config logi-composer.default.migration-tool.dir
logi-composer.default.migration-tool.dir=/dr01/qadapps/systest/servers/logi-composer/default
/migration-tool
```

After you open this file in a text editor, you see a record of all Logi Platform Services objects migrated to Composer successfully, with errors and omissions listed at the end, as in the following example:

```
{
 "objects": [
{
   "sourceId": "12db6b5d-cab1-4b59-ba82-3501c4b5ac15",
   "targetId": "12db6b5d-cab1-4b59-ba82-3501c4b5ac15",
   "sourceName": "Planning Action Messages by Production Line and Site - quality-app app -
planning-app app",
   "targetName": "Planning Action Messages by Production Line and Site - quality-app app -
planning-app app",
   "sourceObjectType": "enrichment",
   "targetObjectType": "source",
   "sourceConnectionId": "eaa62846-4357-420b-858e-a52c38e77de8"
},{
   "sourceId": "vc3c3a44d-dddb-4d56-80f0-5103706b784e",
   "targetId": "c3c3a44d-dddb-4d56-80f0-5103706b784e",
   "sourceName": "Action Message Summary for Production Lines and Sites - quality-app app -
planning-app app",
   "targetName": "Action Message Summary for Production Lines and Sites - quality-app app -
planning-app app",
   "sourceObjectType": "table",
   "targetObjectType": "Raw Data Table",
   "sourceConnectionId": "eaa62846-4357-420b-858e-a52c38e77de8",
   "sourceEnrichmentId": "12db6b5d-cab1-4b59-ba82-3501c4b5ac15"
},{
...
}],
"errors": {
 "objectsMigration": [
 {
   "objectId": "99841643-db5c-d4a4-5814-d68ae0dee15a",
   "objectType": "source based on enrichment and reference",
   "objectName": "",
   "message": "Can't migrate source object based on the enrichment with id - '99841643-db5c-d4a4-
5814-d68ae0dee15a'. Can't execute post request for source object based on the enrichment with id
```

```
- '99841643-db5c-d4a4-5814-d68ae0dee15a', name - 'Cash Flow Analysis'"
},{
   "objectId": "dashboard-0f344b2ca09dc98933e401b7370008f6",
   "objectType": "dashboard",
   "objectName": "Financial Analysis",
   "message": "Can't migrate dashboard with id - dashboard-0f344b2ca09dc98933e401b7370008f6 name
- Financial Analysis. Request failed with status code 400"
},{
...
}],
 "fatal": ""
 }
 }
```

Investigate appendResult File Errors

Review and investigate each of the error entries in the `appendResult` file, which can vary widely for different sets of Action Centers and visuals. This section covers the most common kinds of error, with root causes and instructions on fixing them.

While the `appendResults` file lists dashboards (Action Centers), visuals, and dataviews (KPIs) that were not migrated, for some reason, error details will often be shown only in the `yab.log` file in the section for the `logi-composer-default-lps-migrate-mode-append` YAB command. The remainder of this section assumes that you can open and review both files, as needed.

Fix Dashboards Not Migrated

Errors, like the following, indicate that a dashboard (Action Center) could not be migrated to Composer.

```
{
   "objectId": "dashboard-0f344b2ca09dc98933e401b7370008f6",
   "objectType": "dashboard",
   "objectName": "Financial Analysis",
   "message": "Can't migrate dashboard with id - dashboard-0f344b2ca09dc98933e401b7370008f6 name
- Financial Analysis. Request failed with status code 400"
}
```

Cause

Often, the dashboard could not be migrated because of an error in one or more of the dataviews (KPIs) providing data to visuals in that dashboard. In this case, the `appendResults` file will also reference the dataviews that were not migrated, but will not identify the visuals where the error was raised.

Solution

1. Find the error in the `appendResults` file referencing the dataview used by the skipped dashboard. Usually, the name of the dataview (KPI) will allow you to determine whether that dataview is used on the missing Action Center. The following is a sample error of this kind for the Cash Flow Analysis KPI.

```
{
   "objectId": "99841643-db5c-d4a4-5814-d68ae0dee15a",
   "objectType": "source based on enrichment and reference",
   "objectName": "",
   "message": "Can't migrate source object based on the enrichment with id - '99841643-db5c-d4a4-
5814-d68ae0dee15a'. Can't execute post request for source object based on the enrichment with id
- '99841643-db5c-d4a4-5814-d68ae0dee15a', name - 'Cash Flow Analysis'"
}
```

2. Diagnose and fix the dataview problem, as described below in the Fix Enrichment Dataviews Not Migrated section.

3. After a fix has been applied to the dataview (KPI), re-run the migration tool for the unmigrated dataview object only, as described in the Rerun the Migration for Selected Dataviews section of this document. When the dataview migration succeeds, the visuals and Action Centers that use it will be migrated automatically. Alternatively, the migration tool may be rerun unconditionally with the following YAB command, although throughput time will be greater and log output more verbose:

```
yab logi-composer-default-lps-migrate-mode-append
```

Fix Enrichment Dataviews Not Migrated

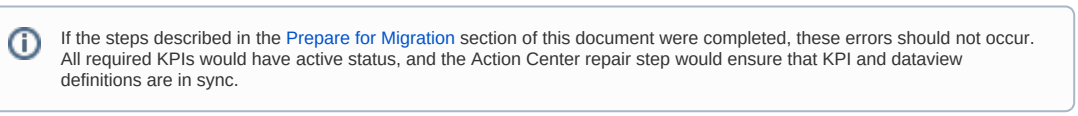
The most common cases where Logi Platform Services objects could not be migrated to Composer involve **enrichment dataviews**, which are associated with the KPIs maintained in Web UI. Enrichment dataviews describe the fields included in a KPI, including labels, formats, and data types. When a dataview cannot be migrated, visuals and dashboards (Action Centers) using that dataview are also not migrated, so the impact of an omitted dataview on the overall migration can be significant. The following is a sample error of this kind:

```
{
   "objectId": "d728f0de-ccca-5faa-9214-8782108784b6",
   "objectType": "source based on enrichment and reference",
   "objectName": "",
   "message": "Can't migrate source object based on the enrichment with id - 'd728f0de-ccca-5faa-
9214-8782108784b6'. Request failed with status code 400"
}
```

Cause

There are several possible causes of an unmigrated dataview, both of which are preventable:

- The KPI associated with the dataview has inactive status in Web UI.
- The fields in the KPI definition do not agree with the dataview definition in Logi Platform Services.



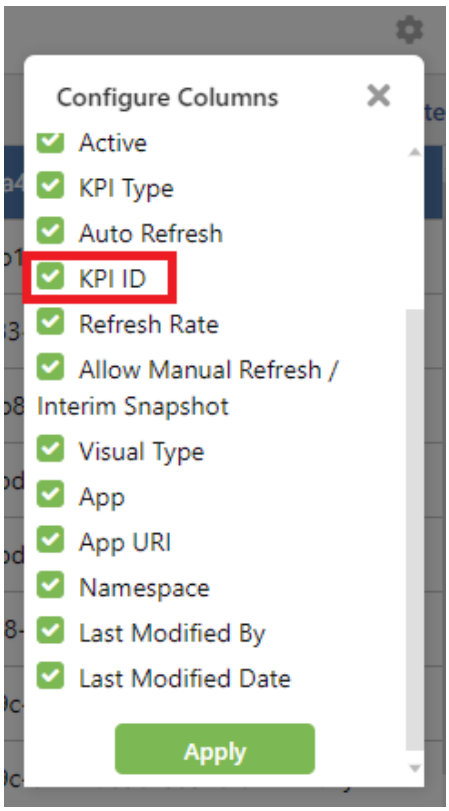
If the steps described in the Prepare for Migration section of this document were completed, these errors should not occur. All required KPIs would have active status, and the Action Center repair step would ensure that KPI and dataview definitions are in sync.

Solution

1. If the name of the KPI associated with the dataview is not included in the error message, find the KPI in the Web UI through its ID value.

    a. Open the KPIs screen in Web UI.

    b. Open the Browse Configuration control and check the KPI ID field in the list, then click Apply.



c. Search for the KPI whose KPI ID value is equal to the `objectId` field from the error message in the `appendResult` file ("d728f0de-ccca-5faa-9214-8782108784b6" in the above example), and select it.

2. Check in the Web UI KPIs screen if the KPI associated with the unmigrated dataview is active. If it should be migrated to Composer, but does not have active status, do the following.

a. Make the KPI active in the KPIs screen of the Web UI by checking the Active checkbox, and saving the KPI.

b. Rerun the migration process for the missing dataview only, as described in the Rerun the Migration Tool for Selected Dataviews section of this document.

3. If the KPI is already active, review the `yab.log` file for a more detailed error related to the dataview by searching for its `objectId` value. In particular, look for a long error message related to an invalid data entity or query, as in the following example:

```
...
[2022-08-24 21:06:03.495][info] - Start preparing source object that related to enrichment
'd728f0de-ccca-5faa-9214-8782108784b6' for migration;
[2022-08-24 21:06:03.495][debug] - Connecting to https://vmlfwy0005.qad.com:22192. The URL - /api
/platform/system.dataviews.enrichment/d728f0de-ccca-5faa-9214-8782108784b6;
[2022-08-24 21:06:03.541][debug] - Successfully connected to https://vmlfwy0005.qad.com:22192.
The URL - /api/platform/system.dataviews.enrichment/d728f0de-ccca-5faa-9214-8782108784b6;
[2022-08-24 21:06:03.541][info] - Start enrichment migration;
[2022-08-24 21:06:03.541][info] - Checking dependent objects;
[2022-08-24 21:06:03.541][debug] - Connecting to https://vmlfwy0005.qad.com:22192. The URL - /api
/platform/system.dataviews.reference/Reference-d728f0de-ccca-5faa-9214-8782108784b6;
[2022-08-24 21:06:03.588][debug] - Successfully connected to https://vmlfwy0005.qad.com:22192.
The URL - /api/platform/system.dataviews.reference/Reference-d728f0de-ccca-5faa-9214-8782108784b6;
[2022-08-24 21:06:03.591][debug] - Connecting to https://vmlfwy0005.qad.com:22131. The URL -
/composer/api/sources/data-entities/describe;
[2022-08-24 21:06:06.922][error] - Error response from - https://vmlfwy0005.qad.com:22131. The
URL - /composer/api/sources/data-entities/describe;
[2022-08-24 21:06:06.922][error] - "Data entity is not valid. org.apache.hive.service.cli.
HiveSQLException: Error running query: org.apache.spark.sql.AnalysisException: Table or view not
found: ...
...
```

This kind of error indicates that the KPI definition in AUX and the dataview definition in Logi Platform Services are out of sync for some reason, possibly because of KPI fields missing from the dataview. In this case, Composer must be temporarily disabled in the environment so that an Action Center repair can be run. Then, Composer must be re-enabled and the migration retried for the repaired dataview. Complete the following steps in this case:

1. Stop the Tomcat Web UI:

```
yab tomcat-webui-stop
```

2. Set the property `qad-analytics-core.composer.enabled` to `false` in the `build/config/configuration.properties` file:

```
qad-analytics-core.composer.enabled=false
```

3. Reconfigure AUX for the property change:

```
yab reconfigure
```

4. Start the Tomcat Web UI:

```
yab tomcat-webui-start
```

5. Run the Action Center repair:

```
yab action-center-dashboard-repair
```

6. Stop the Tomcat Web UI:

```
yab tomcat-webui-stop
```

7. Remove the property assignment for `qad-analytics-core.composer.enabled` in the `build/config/configuration.properties` file, to restore it to its default value.

8. Reconfigure AUX for the property change:

```
yab reconfigure
```

9. Start the Tomcat Web UI:

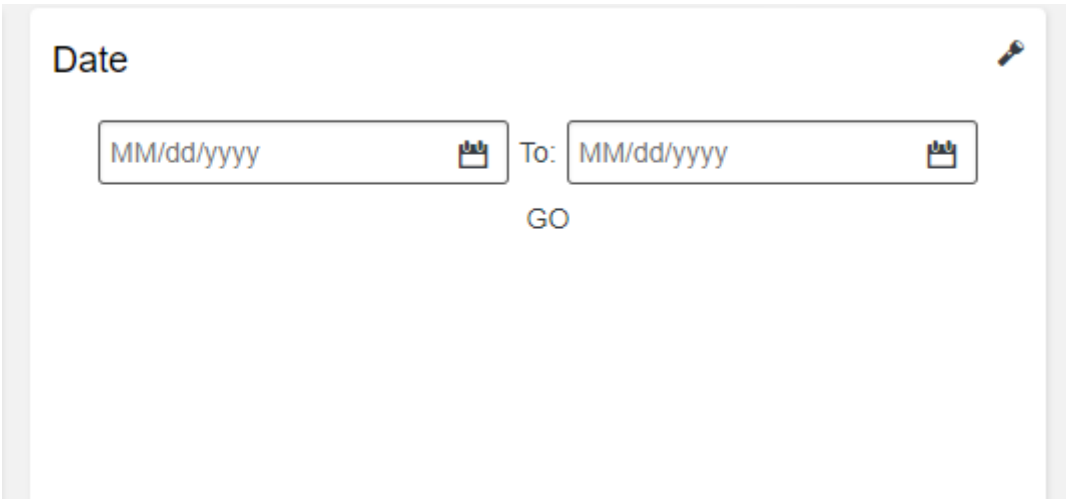
```
yab tomcat-webui-start
```

10. Rerun the migration process for the missing dataview only, as described in Special Migration Procedures. Alternatively, you can rerun the migration tool unconditionally with the following YAB command, although throughput time will be greater and log output more verbose.

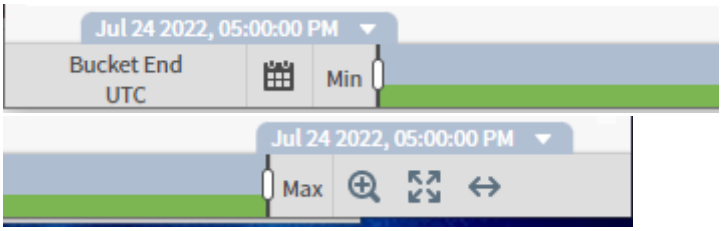
```
yab logi-composer-default-lps-migrate-mode-append
```

Set Timebars in Action Centers for Missing Date Selection Widgets

One major difference between Logi Platform Services and Logi Composer is the support for selected dates and date ranges in Action Centers and visuals. In Logi Platform Services, many Action Centers include widgets used to select key dates or date ranges that filter the displayed data, as in the following example:



In Composer, date select widgets like this do not exist. Instead, Composer provides a built-in timebar control that can be used to filter the data on any visual or the entire Action Center, based on any date field within a KPI. At the Action Center level, this timebar can be controlled near the bottom of the display.



...

Therefore, date widgets, like the one above, are intentionally skipped by the migration process.

To apply default date filtering to an Action Center that formerly used a date select widget, open the Action Center in Web UI and set the from-to dates on the timebar to the desired dates, and save the changes.

Verify That All Missing Action Centers and Visuals Are Migrated

After the reported errors from the `appendResult` file have been fixed and the timebars for all Action Centers have been configured in place of the date select widgets in Logi Platform Services, all missing Action Centers and visuals should be accounted for. Verify that this is the case, and go to the Review and Repair Migration Errors section of this document to investigate various errors that may require rework inside particular KPIs, visuals, or Action Centers.

Review and Repair Migration Errors

This section describes various kinds of errors that can be raised during the migration process, typically because of calculations or filters defined in Logi Platform Services that could not be converted to Logi Composer. In these cases, the Action Centers and visuals will be present in Composer, but specific calculated fields or filters within them might be missing. The specific causes and corrective actions are different for each case.

Review Errors in the YAB Log

While the `appendResult` file used in the previous section of this document lists dataviews (KPIs), dashboards (Action Centers), and visuals that could not be migrated, more detailed errors, in particular filters or calculations, are written to the YAB log file. Open this file in a text editor and find the migration details among the log entries for the `logi-composer-default-lps-migrate-mode-append` YAB process.

```
  2022-09-12T19:14:46,325 DEBUG [main:d53c] APPLY - logi-composer-default-lps-migrate-mode-append
  2022-09-12T19:14:46,338 DEBUG [main:d53c] LogiComposerMigrateProcess - Deploying and Updating
Configuration file [/dr01/qadapps/systest/servers/logi-composer/default/migration-tool/migration.
json]
  2022-09-12T19:14:47,203 DEBUG [main:d53c] HttpCallCommand - GET https://vmlfwy0005.qad.com:22011
/qad-central/api/analytics/composer/migration/config HTTP/1.1
  2022-09-12T19:14:48,278 DEBUG [main:d53c] HttpCallCommand - HttpResponseProxy{HTTP/1.1 200
[Set-Cookie: JSESSIONID=74251BEAA637090D2212EE05F9271795; Path=/qad-central; Secure; HttpOnly,
Vary: Origin, Vary: Access-Control-Request-Metho
  d, Vary: Access-Control-Request-Headers, Cache-Control: no-store, X-Frame-Options: SAMEORIGIN,
Content-Type: application/com.qad.webshell.proxy+json, Transfer-Encoding: chunked, Date: Tue, 13
Sep 2022 02:14:48 GMT, Keep-Alive: timeout=
  60, Connection: keep-alive] ResponseEntityProxy{[Content-Type: application/com.qad.webshell.
proxy+json,Chunked: true]}}
  2022-09-12T19:14:49,151 DEBUG [main:d53c] TemplateEngine - Generated the file [/tmp
/5374721845329519474/3799952944566712977] from the template [/home/mfg/migration-template-
enrichment-dataview.json].
  2022-09-12T19:14:49,181 DEBUG [main:d53c] LogiComposerMigrateProcess - Deploying and Updating
append file [/dr01/qadapps/systest/servers/logi-composer/default/migration-tool/appendFile.json]
  2022-09-12T19:14:49,191 DEBUG [main:d53c] TemplateEngine - Generated the file [/tmp
/5374721845329519474/5306613748331976668] from the template [com/qad/yab/logicomposer/appendfile-
template.json].
  2022-09-12T19:14:49,199 DEBUG [main:d53c] NodejsCommand - /dr01/qadapps/systest/build/catalog
/packages/nodejs/8/11/4/0/bin/node /dr01/qadapps/systest/servers/logi-composer/default/migration-
tool/migrationTool.js -c /dr01/qadapps/systes
  t/servers/logi-composer/default/migration-tool/migration.json
  2022-09-12T19:14:50,788 DEBUG [Thread-3:] STDOUT - [2022-09-13 02:14:50.783][info] - Migration
Tool v.1.112;
...
  2022-09-12T19:17:37,518 DEBUG [Thread-3:] STDOUT - [2022-09-13 02:17:37.518][info] - Result
file 'appendResult1663035457509.json' was created successfully.;
  2022-09-12T19:17:37,545 DEBUG [Thread-3:] STDOUT - [2022-09-13 02:17:37.544][info] - Appended
finished!;
  [2022-09-13 02:17:37.545][info] - Process was done!;
  2022-09-12T19:17:37,590 DEBUG [main:d53c] APPLY - logi-composer-default-lps-migrate-mode-append
UPDATED
```

Search this portion of the file for all occurrences of the string `[error]` to skip all log entries, except for those requiring attention. The rest of this section describes the most important and common errors, and how to address them. Note that some of the errors are false positives that may be ignored or require only a rerun of the migration process with no changes.

Fix Field Calculation Expression Errors

Some calculation formulas used to create visuals in Logi Platform Services will not work in Composer and are not converted automatically to Composer syntax. In these cases, the calculation expression used to create a "derived field" in Composer needs to be re-written in Composer using the correct syntax. Of all the errors raised during migration, this is typically the most common.

The following are examples of calculation errors from the log. Each error shows the source calculation from Logi Platform Services and the invalid converted value ("prepared calculation"):

```
[2022-08-24 20:48:41.040][debug] - Connecting to https://vmlfwy0005.qad.com:22131. The URL -
/composer/api/sources/9c655507-61be-c2bd-5514-137df89757dd/fields;
[2022-08-24 20:48:41.230][error] - Error response from - https://vmlfwy0005.qad.com:22131. The
URL - /composer/api/sources/9c655507-61be-c2bd-5514-137df89757dd/fields;
[2022-08-24 20:48:41.230][error] - "Failed to validate expression: (RIGHT(Start1, 3)) / 60. Can't
find function 'DIV' that accepts parameters of types: [TEXT, NUMBER]. Available functions: DIV
```

```
(dividend:NUMBER, divider:NUMBER):NUMBER, :DIV(operand1:NUMBER,operand2:NUMBER)";
[2022-08-24 20:48:41.230][error] - Source calculation - (RIGHT([Start1],3))/60;
[2022-08-24 20:48:41.230][error] - Prepared calculation - (RIGHT(Start1, 3)) / 60;
[2022-08-24 20:48:41.230][error] - Can't migrate calculation. It will be migrated as stub (you
can fix it manually);
```

```
[2022-08-24 20:48:43.341][debug] - Connecting to https://vmlfwy0005.qad.com:22131. The URL -
/composer/api/sources/9c655507-61be-c2bd-5514-137df89757dd/fields;
[2022-08-24 20:48:43.485][error] - Error response from - https://vmlfwy0005.qad.com:22131. The
URL - /composer/api/sources/9c655507-61be-c2bd-5514-137df89757dd/fields;
[2022-08-24 20:48:43.485][error] - "Failed to validate expression: Elapsed_Time_from_Midnight_EST
- Shift_1_Start_Hour - Shift_1_Start_Hour_Fraction. Can't find function 'SUB' that accepts
parameters of types: [NUMBER, TEXT]. Available functions: SUB(minuend:NUMBER, subtrahend:NUMBER):
NUMBER, :SUB(operand1:NUMBER,operand2:NUMBER)";
[2022-08-24 20:48:43.486][error] - Source calculation - [Elapsed Time from Midnight EST]-[Shift 1
Start Hour]-[Shift 1 Start Hour Fraction];
[2022-08-24 20:48:43.486][error] - Prepared calculation - Elapsed_Time_from_Midnight_EST -
Shift_1_Start_Hour - Shift_1_Start_Hour_Fraction;
[2022-08-24 20:48:43.486][error] - Can't migrate calculation. It will be migrated as stub (you
can fix it manually);
```

```
[2022-08-24 21:13:59.370][debug] - Connecting to https://vmlfwy0005.qad.com:22131. The URL -
/composer/api/sources/ee36242d-68e4-7f9c-5614-94fa78a4a48b/fields;
[2022-08-24 21:13:59.559][error] - Error response from - https://vmlfwy0005.qad.com:22131. The
URL - /composer/api/sources/ee36242d-68e4-7f9c-5614-94fa78a4a48b/fields;
[2022-08-24 21:13:59.560][error] - "Failed to validate expression: CASE WHEN status_code = 'O'
THEN 'Open' ELSE CASE WHEN status_code = 'C' THEN 'Closed' ELSE CASE WHEN status_code = 'S' THEN
'Scheduled' ELSE 'Canceled' END END, 'New' END. Incorrect grammar for provided expression. Refer
to Calculation Help for additional information.";
[2022-08-24 21:13:59.560][error] - Source calculation - IIF([status_code]=="O","Open",
        IIF([status_code]=="C","Closed",IIF([status_code]=="S","Scheduled","Canceled")),"New");
[2022-08-24 21:13:59.560][error] - Prepared calculation - CASE WHEN status_code = 'O' THEN 'Open'
ELSE CASE WHEN status_code = 'C' THEN 'Closed' ELSE CASE WHEN status_code = 'S' THEN 'Scheduled'
ELSE 'Canceled' END END, 'New' END;
[2022-08-24 21:13:59.560][error] - Can't migrate calculation. It will be migrated as stub (you
can fix it manually);
```

Cause

The calculation expression for a calculated field from a visual in Logi Platform Services cannot be converted automatically to the syntax required for a "derived field" in Composer. This may be caused by a built-in Logi Platform Services function that does not exist in Composer. It may also be caused by a conditional formula ('IIF' function) from Logi Platform Services that was not correctly converted to a CASE statement, as required for Composer.

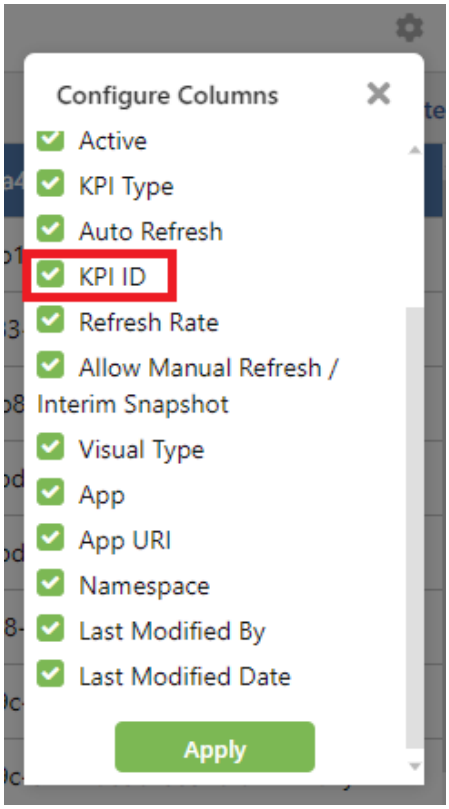
Solution

Find the definition of the derived field in Composer, that corresponds to the calculated field from Logi Platform Services, and write a new calculation expression for it. In Composer, derived fields are stored in the "source" (KPI) object, not in the visual, as was usually the case with Logi Platform Services. Therefore, the first step is to find the KPI where the invalid calculation is defined.

1. Find the KPI in the Web UI through its KPI ID value, which is the same as the source ID in Composer.

   a. Open the KPIs screen in Web UI.

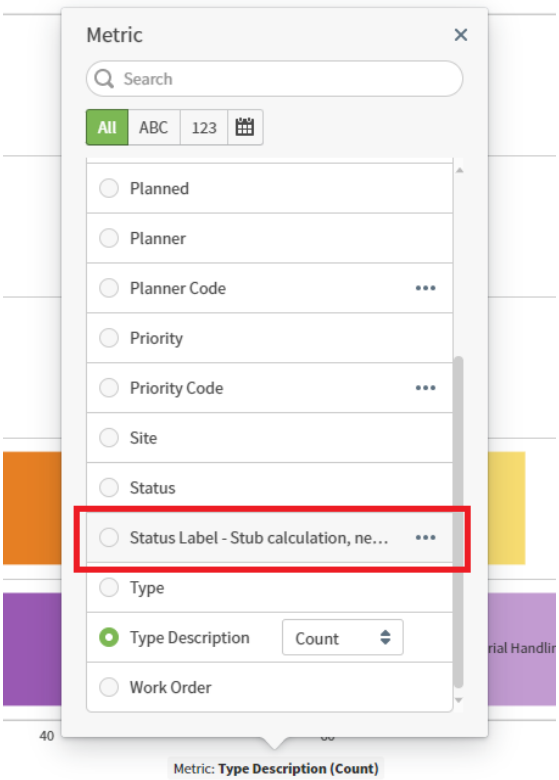
   b. Open the Browse Configuration control and check the KPI ID field in the list, then click Apply.



2. Search for the KPI whose KPI ID value is equal to the UUID portion of the URL referenced in the error message from the log ("ee36242d-68e4-7f9c-5614-94fa78a4a48b" in the third example above), and select it.

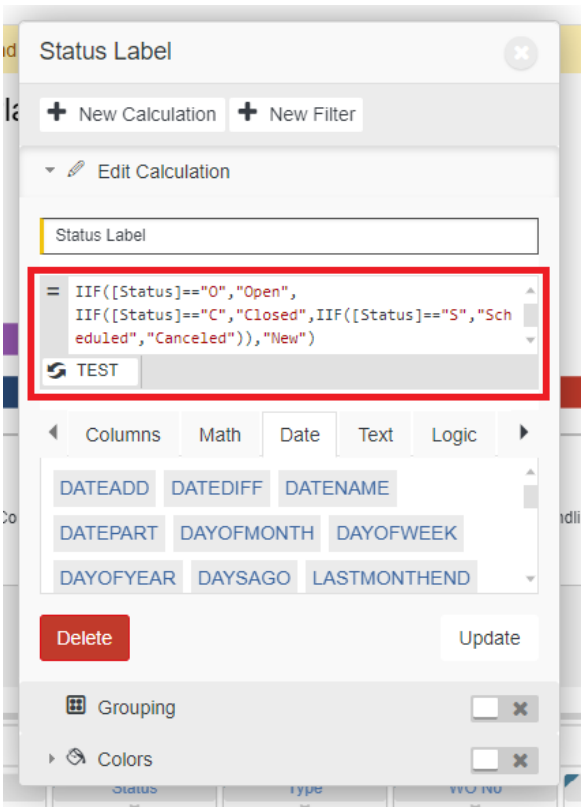
a. Go to the Visuals panel of the screen, and open any one of the visuals for the selected KPI.

b. Select any of the fields used for the axes of the chart to display a list of all the fields. Find the field with the invalid calculation, whose name ends in the string "- Stub calculation, need to be reviewed and fixed manually."



c. Click the three-dot control ("...") next to the field, and select Edit in the displayed pop-up dialog. The calculation string will be blank.

3. Re-implement the calculation using valid Composer syntax. It may be helpful to review the original calculation in a non-Composer AUX environment for comparison purposes.



4. Click Save to save changes to the derived field.

Fix Unsupported Filter Expressions

Some filters defined inside visuals in Logi Platform Services use built-in functions that do not exist in Composer. If the filter is still needed in Composer, it must be defined there. The following are examples of this error from the log:

```
[2022-08-24 21:29:57.250][debug] - Connecting to https://vmlfwy0005.qad.com:22131. The URL -
/composer/api/sources/a588c665-88e9-4a98-5414-4d96f8302610/fields;
```

```
[2022-08-24 21:29:57.372][debug] - Successfully connected to https://vmlfwy0005.qad.com:22131.
The URL - /composer/api/sources/a588c665-88e9-4a98-5414-4d96f8302610/fields;
[2022-08-24 21:29:57.372][debug] - Processing Old Composer filter expression - ' (TIMESTAMP
([due_date]) >= 'Invalid date' AND TIMESTAMP([due_date])  <=  'Invalid date') ';
[2022-08-24 21:29:57.379][error] - Can't migrate filters. The object will be migrated without
filters;
[2022-08-24 21:29:57.380][error] - Unsupported date function for filter ''Invalid date'';
```

Cause

Functions used in the filter are not supported in Composer.

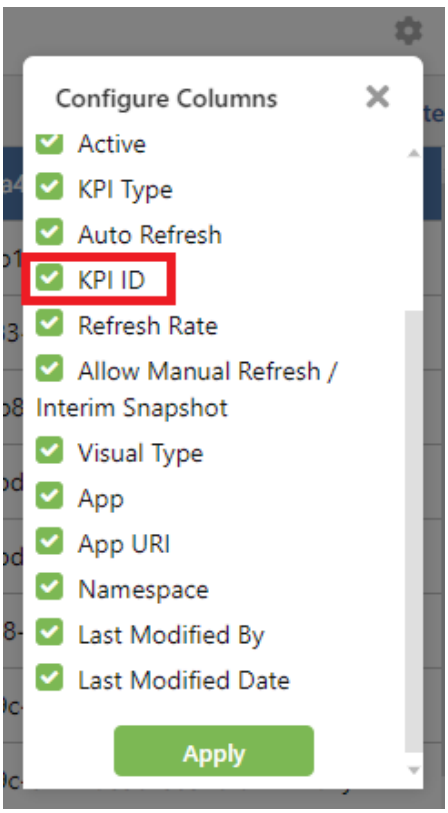
Solution

Find the definition of the filter in Composer, corresponding to the one from Logi Platform Services. Determine if the filter definition is needed and, if so, re-implement it. The log file does not identify the visual(s) where the filter is used, only the source (KPI). Therefore, the first step is to find the KPI and its visual(s) where the filter is needed.

1. Find the KPI in the Web UI through its KPI ID value, which is the same as the source ID in Composer.

    a. Open the KPIs screen in Web UI.

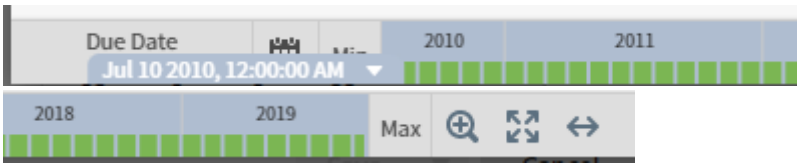
    b. Open the Browse Configuration control and check the KPI ID field in the list, then click Apply.



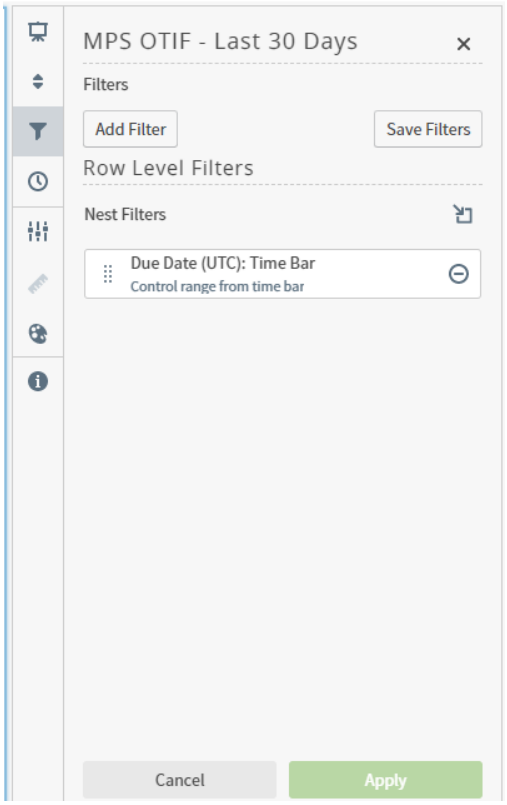
2. Search for the KPI whose KPI ID value is equal to the UUID portion of the URL referenced in the error message from the log ("a588c665-88e9-4a98-5414-4d96f8302610" in the above example), and select it.

3. Go to the Visuals panel of the screen, and open each of the visuals for the selected KPI. For comparison purposes, it may be helpful to open the same visuals in a non-Composer AUX environment.

4. Determine whether the unmigrated filter is needed in Composer. If the filter is being applied to date fields, as in the above example, it does not have to be created in Composer because Composer supports date and date range filtering using its built-in timebar feature. The timebar within the visual can be enabled and configured at the bottom of the Visual Builder window.



5. If the filter is needed in Composer, define it in the filter area within the right-hand sidebar in the Visual Builder window and click Apply.



6. Click Save to save the changes to the visual.

Connection Error and Retry

You may see connection errors in the YAB log, where the migration process is unable to connect to Logi Platform Services and receives a "null" response. Whenever this error occurs, the migration process will retry the connection several times. The following example shows log entries for this case:

```
[2022-08-24 20:24:04.911][debug] - Connecting to https://vmlfwy0005.qad.com:22192. The URL - /api
/platform/system.connections/eaa62846-4357-420b-858e-a52c38e77de8?
action=computeGraph&$expand=downstream;
[2022-08-24 20:24:07.406][error] - Error response from - https://vmlfwy0005.qad.com:22192. The
URL - /api/platform/system.connections/eaa62846-4357-420b-858e-a52c38e77de8?
action=computeGraph&$expand=downstream;
[2022-08-24 20:24:07.406][error] - "null";
[2022-08-24 20:24:07.406][error] - Can't get enrichments for connection - eaa62846-4357-420b-858e-
a52c38e77de8. Retrying;
[2022-08-24 20:24:07.407][debug] - Try #0;
[2022-08-24 20:25:07.463][debug] - Connecting to https://vmlfwy0005.qad.com:22192. The URL - /api
/platform/system.connections/eaa62846-4357-420b-858e-a52c38e77de8?
action=computeGraph&$expand=downstream;
[2022-08-24 20:25:09.445][error] - Error response from - https://vmlfwy0005.qad.com:22192. The
URL - /api/platform/system.connections/eaa62846-4357-420b-858e-a52c38e77de8?
action=computeGraph&$expand=downstream;
[2022-08-24 20:25:09.445][error] - "null";
```

```
[2022-08-24 20:25:09.445][error] - Try #0 is not successful;
[2022-08-24 20:25:09.446][debug] - Try #1;
[2022-08-24 20:26:09.506][debug] - Connecting to https://vmlfwy0005.qad.com:22192. The URL - /api
/platform/system.connections/eaa62846-4357-420b-858e-a52c38e77de8?
action=computeGraph&$expand=downstream;
[2022-08-24 20:27:05.995][debug] - Successfully connected to https://vmlfwy0005.qad.com:22192.
The URL - /api/platform/system.connections/eaa62846-4357-420b-858e-a52c38e77de8?
action=computeGraph&$expand=downstream;
...
```

Cause

Inability to connect to Logi Platform Services for unknown reasons, often intermittently, can occur for any of the following reasons:

- A heavy server load, causing slow response times from Logi Platform Services.
- Logi Platform Services has not been fully started yet.
- Network problems on the machine such as a blocked port.

Solution

1. If the log shows that the retries succeeded, the problem is solved and nothing else is needed.

2. Otherwise, check that Logi Platform Services is running, and that only a single instance of the Logi Platform Services processes are running:

```
yab logi-platform-services-default-status
```

```
$ pgrep -af logi-platform-services
105721 /dr01/qadapps/systest/build/catalog/packages/nodejs/8/11/4/0/bin/node /dr01/qadapps/systest
/servers/logi-platform-services/default/platform/ds/server/logiDataService.js
105729 /dr01/qadapps/systest/build/catalog/packages/nodejs/8/11/4/0/bin/node /dr01/qadapps/systest
/servers/logi-platform-services/default/platform/web-server/logiApplicationService.js
105876 /dr01/qadapps/systest/build/catalog/packages/nodejs/8/11/4/0/bin/node /dr01/qadapps/systest
/servers/logi-platform-services/default/platform/microservices/microservice.js --host=vmlfwy0005.
qad.com --port=22151 --username=ngpamqadmin --password=zH2^GEVFkc@WNSv --pollingInterval=15 --
logLevel=debug --ignoreHTTPSErrors
105883 /dr01/qadapps/systest/build/catalog/packages/nodejs/8/11/4/0/bin/node /dr01/qadapps/systest
/servers/logi-platform-services/default/platform/microservices/microservice.js --host=vmlfwy0005.
qad.com --port=22151 --username=ngpamqadmin --password=zH2^GEVFkc@WNSv --pollingInterval=15 --
logLevel=debug --ignoreHTTPSErrors
105888 /dr01/qadapps/systest/build/catalog/packages/nodejs/8/11/4/0/bin/node /dr01/qadapps/systest
/servers/logi-platform-services/default/platform/microservices/microservice.js --host=vmlfwy0005.
qad.com --port=22151 --username=ngpamqadmin --password=zH2^GEVFkc@WNSv --pollingInterval=15 --
logLevel=debug --ignoreHTTPSErrors
105896 /tech/java/openjdk1.8.0_322/bin/java -Xms512m -Xmx1024m -javaagent:/dr01/qadapps/systest
/servers/logi-platform-services/default/platform/ds/server/lib/spring-instrument-4.3.3.RELEASE.
jar -Dfile.encoding=UTF-8 -Djava.security.properties=/dr01/qadapps/systest/servers/logi-platform-
services/default/platform/settings/java.security -Dsun.rmi.transport.tcp.maxConnectionThreads=0 -
Dlogi.home=/dr01/qadapps/systest/servers/logi-platform-services/default -Dloader.path=/dr01
/qadapps/systest/servers/logi-platform-services/default/platform/ds-extensions -jar /dr01/qadapps
/systest/servers/logi-platform-services/default/platform/ds/server/target/dataservice-1.0.0.jar
com.logi.ds.dataserver.ApplicationLauncher
```

3. If this is not the case, kill any duplicate processes from the command line and restart Logi Platform Services. After starting Logi Platform Services, wait several minutes to ensure that all processes are running.

```
yab logi-platform-services-default-start
```

4. Rerun the migration process using the following YAB command. As this problem can be intermittent, simply rerunning the process may be sufficient.

```
yab logi-composer-default-lps-migrate-mode-append
```

Field Already Exists

The migration process may try to migrate the same field multiple times, when it is used on multiple visuals. The following is a sample log entry for this case:

```
[2022-08-24 20:28:56.659][debug] - Successfully connected to https://vmlfwy0005.qad.com:22131.
The URL - /composer/api/sources/862404b7-980e-8885-5414-5663a00cee43/fields;
[2022-08-24 20:28:56.660][debug] - Connecting to https://vmlfwy0005.qad.com:22131. The URL -
/composer/api/sources/862404b7-980e-8885-5414-5663a00cee43/fields/record_count;
[2022-08-24 20:28:57.010][error] - Error response from - https://vmlfwy0005.qad.com:22131. The
URL - /composer/api/sources/862404b7-980e-8885-5414-5663a00cee43/fields/record_count;
[2022-08-24 20:28:57.012][error] - "Field with label \"Record Count\" already exists";
[2022-08-24 20:28:57.015][error] - Can't update label for field with name - record_count, from
Record Count (1) to Record Count for source with id - 862404b7-980e-8885-5414-5663a00cee43;
...
```

Cause

A field with the same name already exists in the same KPI.

Solution

No action is needed because an additional copy of the field will cause no problems and will not be visible on the KPI screen.

Visual Already Exists

The migration process may try to migrate the same visual multiple times, when it is used on multiple Action Centers. The following is a sample log entry for this case:

```
[2022-08-24 20:30:12.103][debug] - Connecting to https://vmlfwy0005.qad.com:22131. The URL -
/composer/api/visuals/3b43a07f-c59b-4bd2-a37e-a36d0ae19b14;
[2022-08-24 20:30:12.263][error] - Error response from - https://vmlfwy0005.qad.com:22131. The
URL - /composer/api/visuals/3b43a07f-c59b-4bd2-a37e-a36d0ae19b14;
[2022-08-24 20:30:12.264][error] - "Visual with the name 'OTIF Percent for Last Weeks' already
exists.";
[2022-08-24 20:30:12.264][debug] - Connecting to https://vmlfwy0005.qad.com:22131. The URL -
/composer/api/visuals/;
[2022-08-24 20:30:12.553][debug] - Successfully connected to https://vmlfwy0005.qad.com:22131.
The URL - /composer/api/visuals/;
[2022-08-24 20:30:12.555][warn] - Can't migrate visual, because - 'Visual with the name 'OTIF
Percent for Last Weeks' already exists.'. Retrying to migrate visual with it's name + current
millis;
...
```

Cause

A visual with the same name already exists in the same KPI. The migration process creates a separate visual for each copy of the original visual that was present in Logi Platform Services.

Solution

No action is needed because an additional copy of the field will cause no problems. However, consider removing unnecessary copies as part of a final clean up, as described in the Clean Up Action Centers and Visuals section of this document.

Authentication Error

Authentication errors with subsequent retries, similar to the following example, are sometimes listed in the log file:

```
[2022-08-24 20:36:23.679][debug] - Connecting to https://vmlfwy0005.qad.com:22192. The URL - /api
/platform/system.dataviews.enrichment?action=execute;
[2022-08-24 20:36:23.731][error] - Error response from - https://vmlfwy0005.qad.com:22192. The
URL - /api/platform/system.dataviews.enrichment?action=execute;
[2022-08-24 20:36:23.731][warn] - It's auth related error response. Authenticating and retrying
the request.;
```

Cause

The absense of a valid Logi Platform Services session for unknown reasons, often intermittently.

Solution

No action is needed because the automatic retries almost always resolve the problem before the migration process is disrupted.

Clean Up Action Centers and Visuals

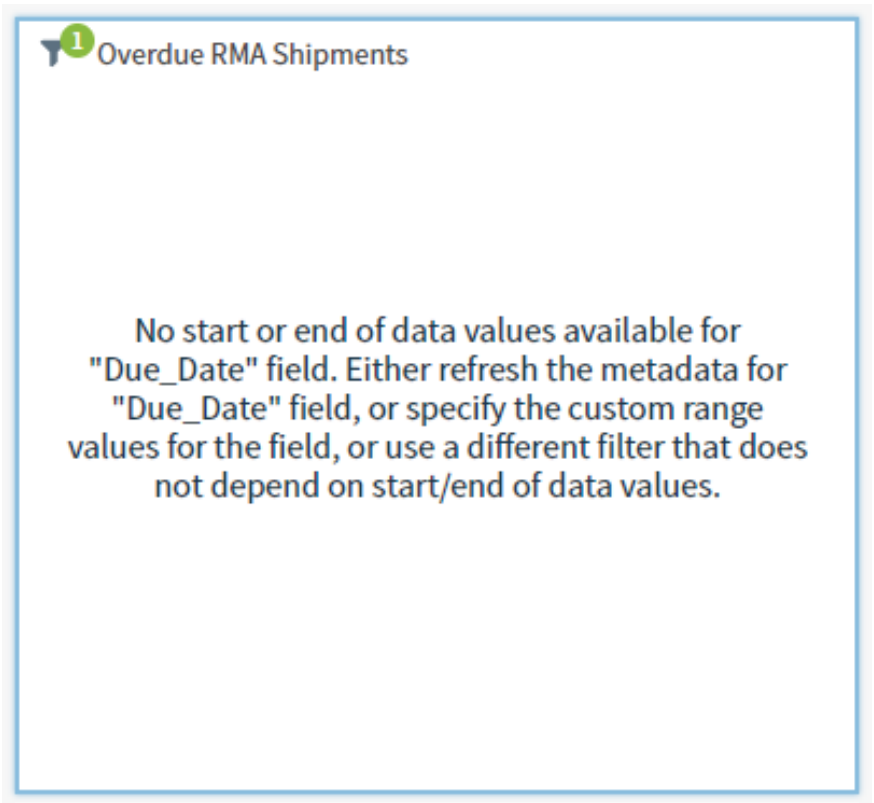
This section describes fixes typically required to migrated Action Centers and visuals to improve their display and to make them easier to maintain. The migrated Action Centers and visuals may function correctly after all errors have been corrected, as described in the previous section of this document, but you can make them more usable and maintainable using the steps in this section.

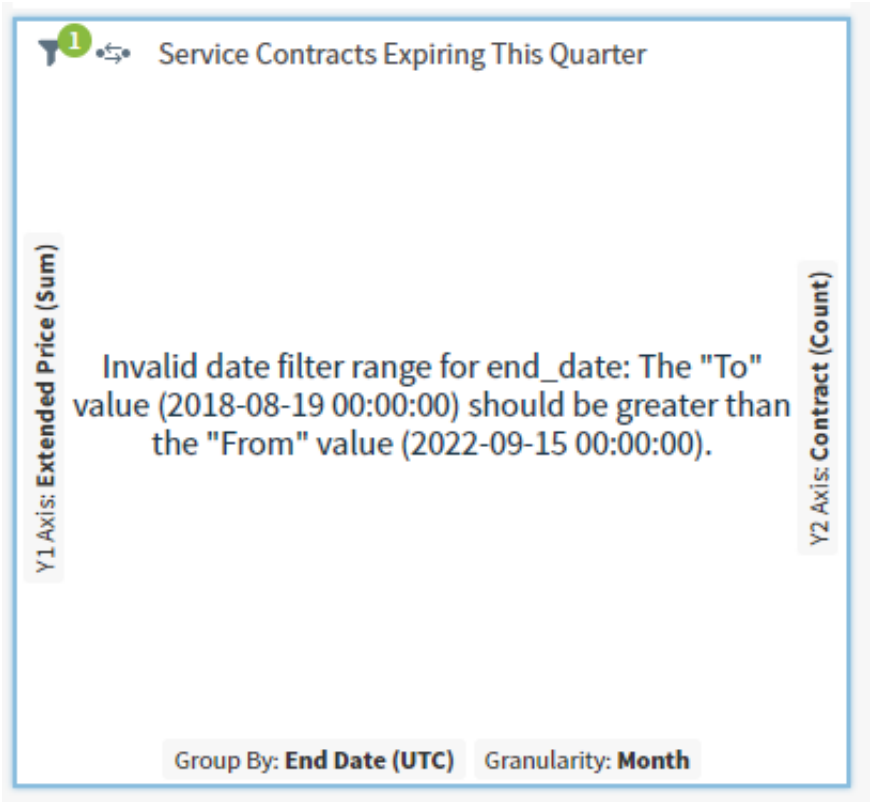
Review Action Centers and Visuals in Web UI

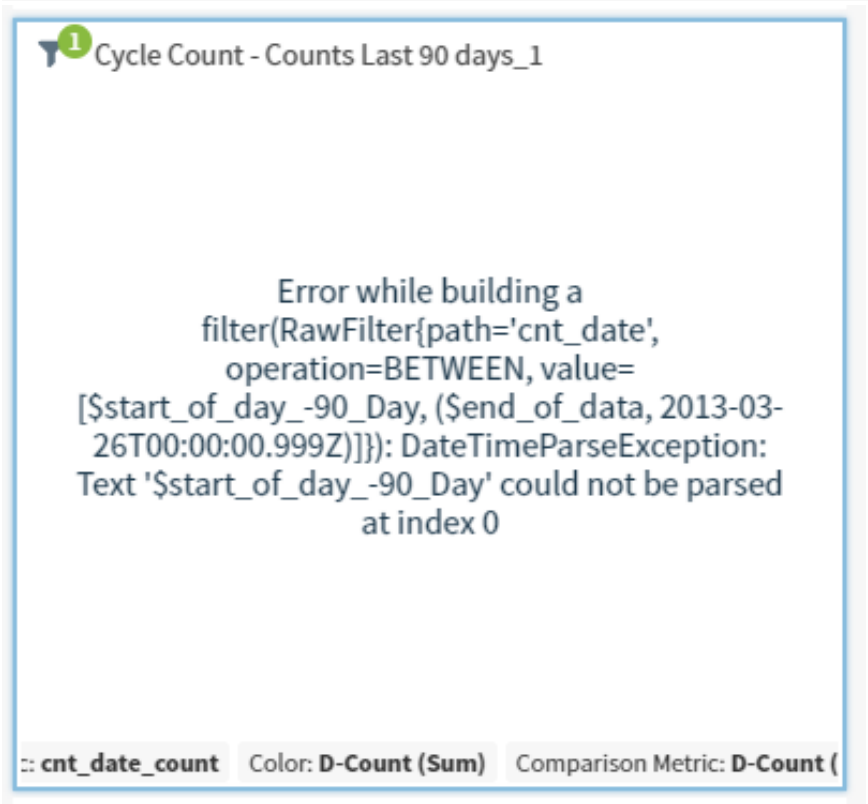
For the steps in this section, no log files are needed. All steps are completed by reviewing and updating each Action Center and visual in the Web UI, making any necessary changes.

Fix Visuals for Date Ranges with No Data Available

When KPIs in the target AUX environment return no data because of the lack of business data in the system, migrated visuals that filter the data based on date ranges sometimes display run-time errors. The following are some sample errors of this kind:



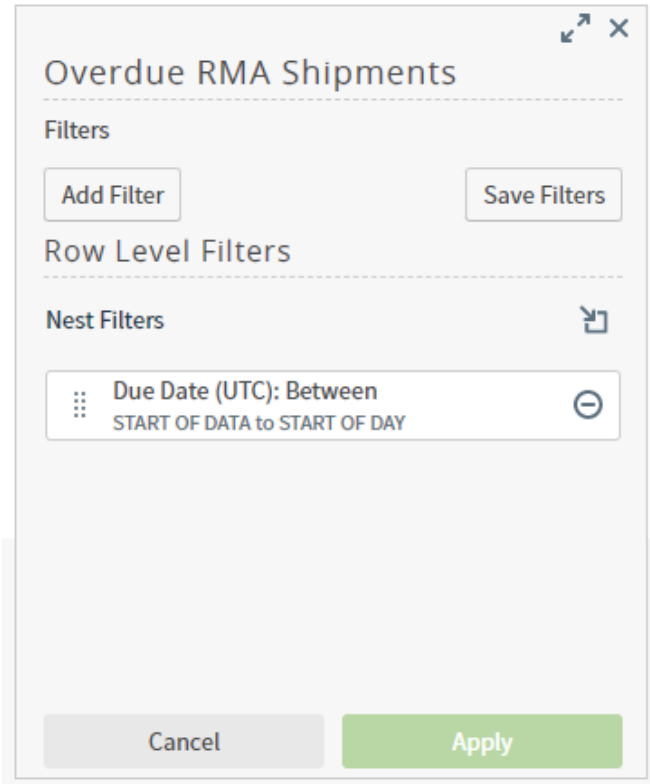


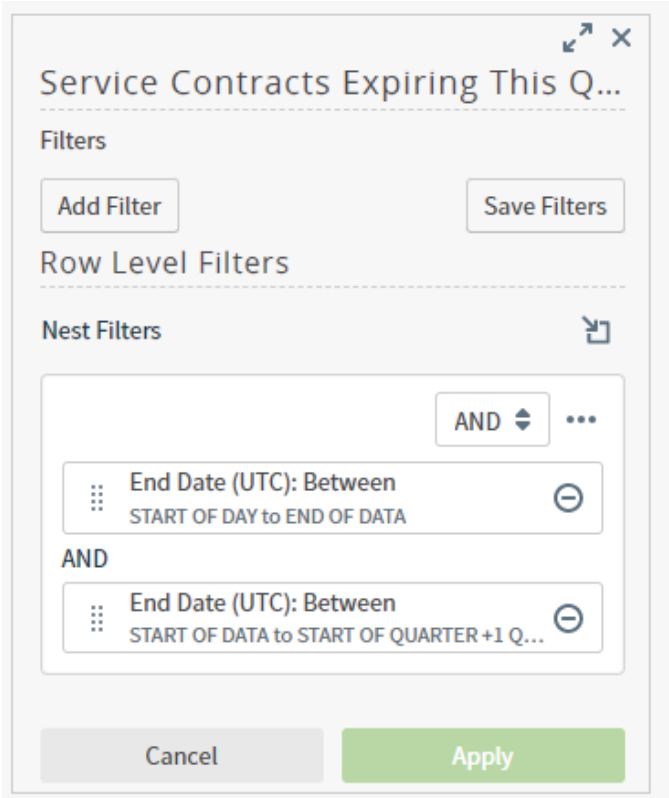


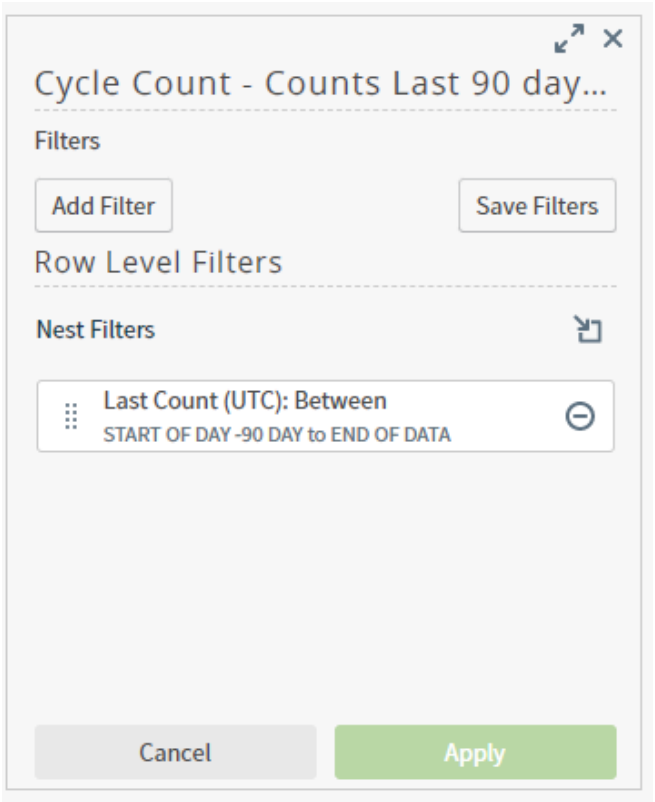
These errors do not appear in log files, but are visible when displaying the visuals where the KPI used by the visual contains no business data to return.

The errors appear only in cases where dynamic date range filters are used in the visual, as opposed to static ranges. "Dynamic" means that the start and/or end of the data range is determined by the dataset being displayed, rather than a boundary based on the calendar such as "1 Jan 2022," "Previous Year-End," "This Quarter," or "2 Weeks Ago." The errors are harmless and disappear automatically when business data is included in the visuals, but you can remove them explicitly by changing the date range filter.

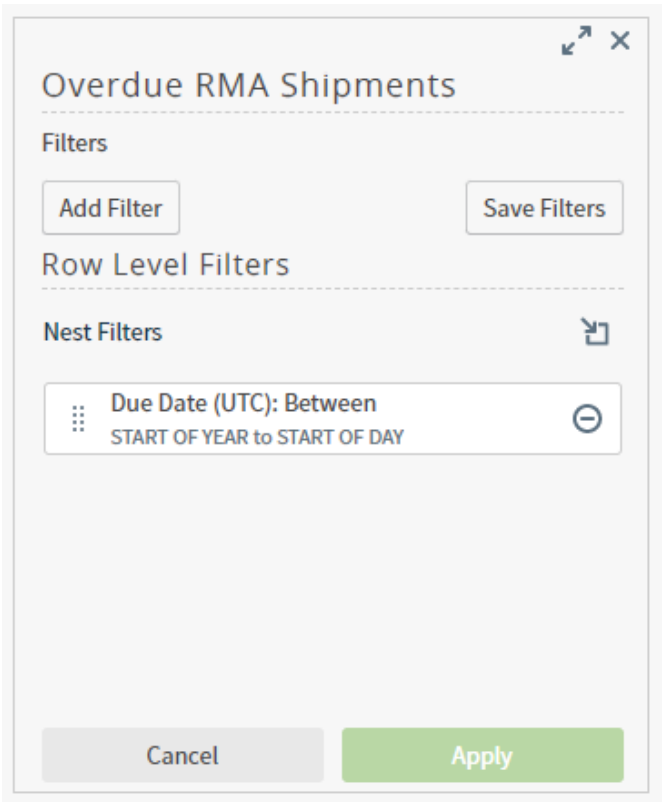
When expanded, the filter for one of these visuals shows the start and/or end date as "START OF DATA" or "END OF DATA."



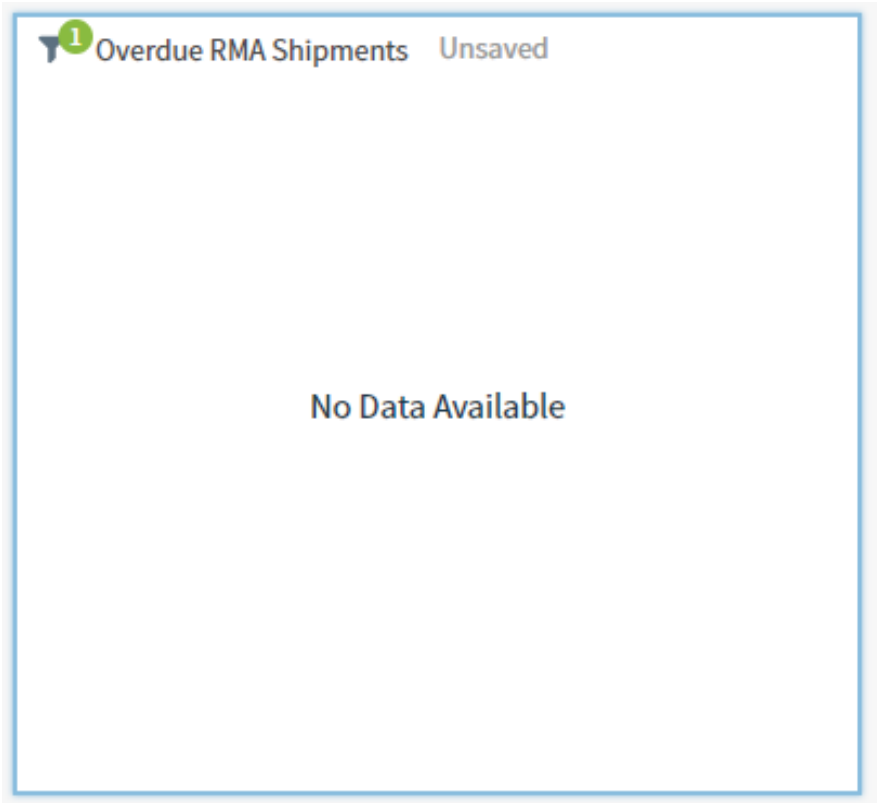




To remove the errors, modify the filter to change "START OF DATA" and "END OF DATA" to some static value, as shown in the following example:



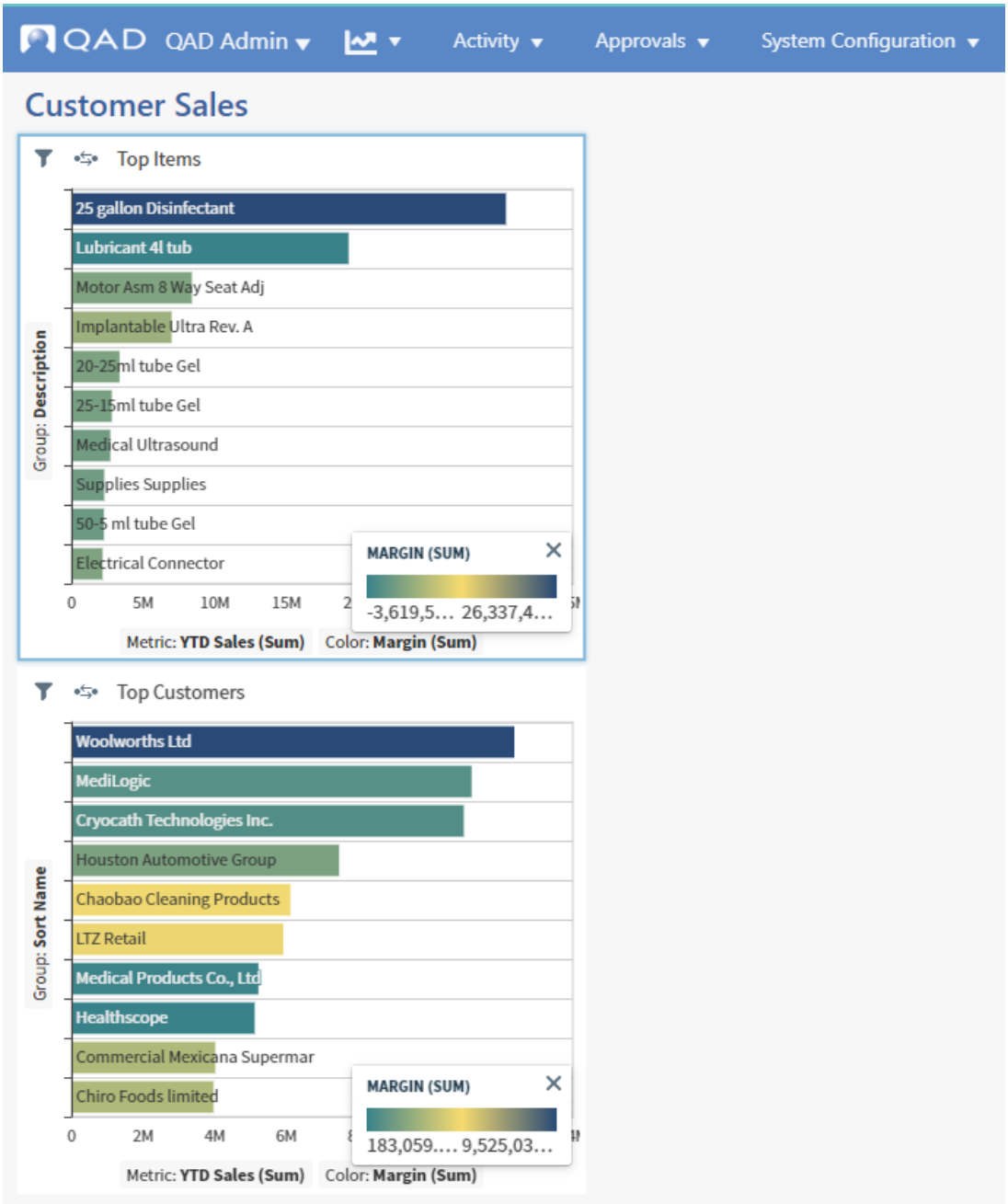
When you modify the filter, the expected "No Data Available" message displays on the changed visual:



However, if the visual requires a dynamic, rather than static, date range; you cannot make this kind of change and the error message remains.

Change Layouts of Migrated Action Centers

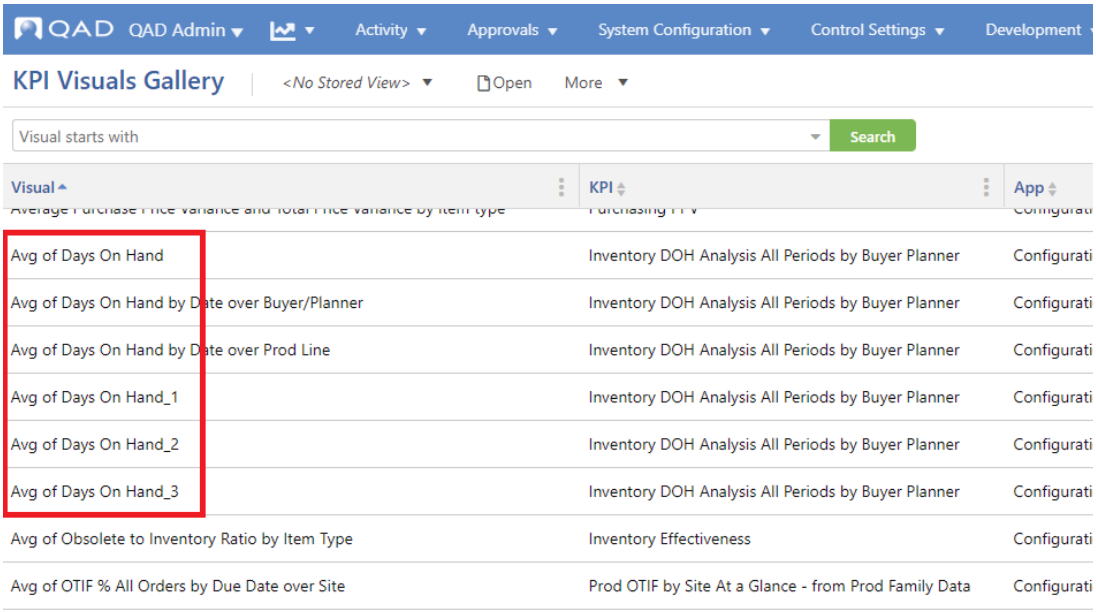
When Action Centers are migrated from Logi Platform Services to Logi Composer, the panels are automatically arranged under a single column, rather than in multiple rows and columns across the page. This is the case because the rendering of the same visuals using Composer is so different, and, in most cases, better, that users will nearly always want to change the layout and sizing of the panels within each Action Center. In Composer, you can resize panels much more flexibly than in Logi Platform Services—you can significantly improve the usability and readability of the visuals in Composer with a few manual adjustments.



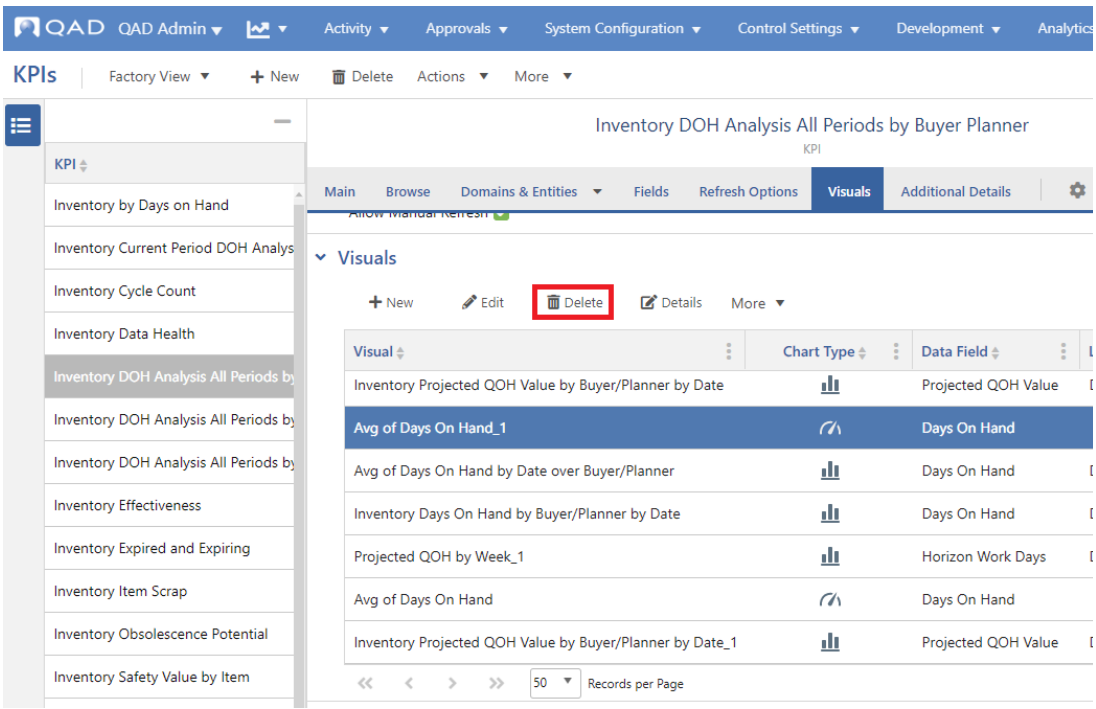
To change the layout and sizing of the panels, open the Action Center in the Web UI and simply drag and expand-contract individual panels, as desired. Save the results.

Delete-Replace Redundant Visuals

The migration process often creates multiple copies of the same visual, assigning a unique numeric suffix to the original name (example: "Select Buyer-Planner," "Select Buyer-Planner_1," "Select Buyer-Planner_2," and so on). This happens because Logi Platform Services contains distinct copies of a visual for each Action Center where that visual is used, as well as the original copy in the Visuals Gallery. Composer, on the other hand, stores only a single common copy of each visual, which you can use in any number of Action Centers. As each copy is converted separately during the migration process, Composer often ends up with redundant copies of visuals. This situation is especially true for selection list widgets (examples: "Select Site," "Select GL Account," "Select Customer Name," and so on), which are often used in multiple Action Centers. You can easily see the duplicates on the KPI Visuals Gallery screen in the Web UI, when the entries are sorted in Visual order.



These duplicate visuals work and you do not have to remove them. However, if common visuals are used across many Action Centers, having multiple copies of the same visual means that all copies must be updated individually whenever a common change is needed. If this is the case, consider modifying the Action Centers that use the common visuals so that they use only one of the copies. When the unnecessary copies have been been removed-replaced in all Action Centers, you can delete those copies by locating the KPI of the visual on the KPIs screen, selecting the specific visual in the Visuals panel, and clicking the Delete button.



Complete the Migration Process

This section describes the final steps to end the migration from Logi Platform Services to Composer, disabling and/or removing AUX components that are no longer needed.

1. Disable the Migration Tool

Run the following command to disable the migration tool used to convert Logi Platform Services objects to Logi Composer:

```
yab logi-composer-migration-complete
```

This command disables all migration processes, stops Logi Platform Services, and disables all YAB commands related to Logi Platform Services. It can be run only once in an AUX environment.

2. Remove Logi Platform Services from the Environment

After the migration to Composer is complete, uninstall Logi Platform Services and remove it from the AUX environment. While this step is not mandatory, it avoids the unnecessary use of system resources when Logi Platform Services is no longer needed. To remove Logi Platform Services, run the following command:

```
yab logi-platform-services-default-remove
```

Once executed, this command cannot be undone or reversed.

Special Migration Procedures

This section is a reference describing several procedures that would not normally be required to migrate from Logi Platform Services to Logi Composer, but may be needed for special upgrade scenarios or to correct particular migration errors. They are referenced from other sections of this document to help resolve some error conditions.

Rerun the Migration for Selected Dataviews

Normally, the migration process is run only once in an AUX environment during the YAB update at the time it is being upgraded to the September 2022 release, or sometime after the upgrade when Logi Composer is enabled. However, it is possible to rerun the migration for selected dataviews (KPIs) in Logi Platform Services, rather than re-migrating every dataview (KPI), dashboard (Action Center), and visual again. This can be useful to retry the migration of a single dataview and its associated Action Centers and visuals that failed the first time because of an environment or data problem.

1. Obtain the IDs of the dataviews in Logi Platform Services to be included in the migration run. The ID values are UUIDs (example: "ee36242d-68e4-7f9c-5614-94fa78a4a48b"), and are generally labeled `objectId` in the migration `appendResult` file, as in the following example:

```
{
    "objectId": "99841643-db5c-d4a4-5814-d68ae0dee15a",
    "objectType": "source based on enrichment and reference",
    "objectName": "",
    "message": "Can't migrate source object based on the enrichment with id - '99841643-db5c-d4a4-
5814-d68ae0dee15a'. Request failed with status code 400"
}
```

2. Make a local copy of the standard migration template file used to configure the scope of the migration process. The pathname of this file is stored in the `logi-composer.default.migration-tool.configuration-template` YAB property.

```
yab config logi-composer.default.migration-tool.configuration-template
```

The contents of the standard migration file are similar to the following:

```
{
    "connection-timeout": "$yab.eval('${instancekey}.connection-timeout')",
    "mode": "${mode}",
    "property-replacing": [{
        "namespace": "system.connections",
        "id": "$yab.eval('sourceId')",
        "properties": [{
            "path": "payload.password",
            "value": "$yab.eval('${instancekey}.connection-pwd')"
        }]
    }],
    "data-points-limit": $yab.eval('${instancekey}.data-point-limit'),
    #if (${mode} == "append")
    "appendFile": "appendFile.json",
    #end
    "source-tags": {
        "namespace": "com.qad.tags",
        "linkTypes": [
            "comQadApp",
            "comQadKpi"
        ],
        "ids": [
            "!urn:app:com.qad.*",
            "!urn:kpi:com.qad.*"
         ]
    },
    "source-url": "$yab.eval('${${instancekey}.lps}.url')",
    "source-user": "$yab.eval('${${instancekey}.lps}.username')",
    "source-pwd": "$yab.eval('${${instancekey}.lps}.password')",
    "target-url": "$yab.eval('${composerinstancekey}.url')",
    "target-user": "$yab.eval('${composerinstancekey}.users.admin.username')",
    "target-pwd": "$yab.eval('${composerinstancekey}.users.admin.password')"
}
```

3. In the local copy, remove the `source-tags` property and replace it with a `source-objects` property that references the dataviews to be migrated in the `ids` property. Do not make any other changes to the file. The following example shows a reconfigured template file with the correct syntax for the `source-objects` property. In this example, the `objectId` of the dataview to be migrated is `99841643-db5c-d4a4-5814-d68ae0dee15a`. You can also specify multiple dataviews by entering a comma-delimited list of `objectId` values inside the `ids` array of this file, instead of a single entry.

```
{
    "connection-timeout": "$yab.eval('${instancekey}.connection-timeout')",
    "mode": "${mode}",
    "property-replacing": [{
        "namespace": "system.connections",
        "id": "$yab.eval('sourceId')",
        "properties": [{
            "path": "payload.password",
            "value": "$yab.eval('${instancekey}.connection-pwd')"
        }]
    }],
    "data-points-limit": $yab.eval('${instancekey}.data-point-limit'),
    #if (${mode} == "append")
    "appendFile": "appendFile.json",
    #end
    "source-objects": [
        {
            "namespace": "system.dataviews.enrichment",
            "ids": [
                "99841643-db5c-d4a4-5814-d68ae0dee15a"
            ]
        }
    ],
    "source-url": "$yab.eval('${${instancekey}.lps}.url')",
    "source-user": "$yab.eval('${${instancekey}.lps}.username')",
    "source-pwd": "$yab.eval('${${instancekey}.lps}.password')",
    "target-url": "$yab.eval('${composerinstancekey}.url')",
    "target-user": "$yab.eval('${composerinstancekey}.users.admin.username')",
    "target-pwd": "$yab.eval('${composerinstancekey}.users.admin.password')"
}
```

4. Rerun the migration process using the new template file through the following YAB command. The new template file will be used only in the context of this command, without permanently changing the value of the `logi-composer.default.migration-tool.configuration-template` YAB property.

```
yab -logi-composer.default.migration-tool.configuration-template:<path to cloned template> logi-composer-default-lps-migrate-mode-append
```

Example:

```
yab -logi-composer.default.migration-tool.configuration-template:/home/mfg/migration-template-enrichment-dataview.json logi-composer-default-lps-migrate-mode-append
```

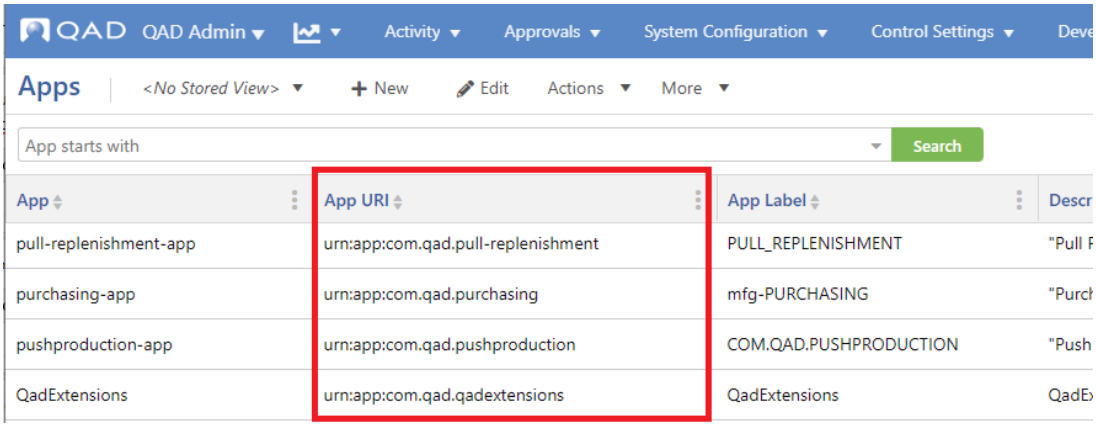
5. Review the results of the migration in the `appendResult` file and YAB log, as described in the earlier sections of this document.

Rerun the Migration for Selected Apps

By default, the migration process attempts to migrate all dataviews (KPIs), dashboards (Action Centers), and visuals from Logi Platform Services to Composer that were not provided by any of the QAD apps. Because all QAD-provided Action Centers for the September 2022 release are installed only as Composer objects, there is no need to migrate them. However, under some circumstances, it may be desirable to configure the migration process to cover only selected non-QAD apps.

For example, an existing AUX customer upgrading to the September 2022 release may have installed a mix of QAD-provided, third-party-provided, and internally developed apps. By default, the third-party and internally developed Action Centers will be migrated automatically to Composer during the upgrade. However, if there is a reason to migrate the third-party Action Centers but not the internally developed ones, or vise versa, it is possible to configure the migration process to filter by app.

1. Identify the URIs of the apps to be included in the migration. The URIs can be found by displaying the Apps screen in Web UI.



2. Make a local copy of the standard migration template file used to configure the scope of the migration process. The pathname of this file is stored in the `logi-composer.default.migration-tool.configuration-template` YAB property.

```
yab config logi-composer.default.migration-tool.configuration-template
```

The contents of the standard migration file is similar to the following:

```
{
    "connection-timeout": "$yab.eval('${instancekey}.connection-timeout')",
    "mode": "${mode}",
    "property-replacing": [{
        "namespace": "system.connections",
        "id": "$yab.eval('sourceId')",
        "properties": [{
            "path": "payload.password",
            "value": "$yab.eval('${instancekey}.connection-pwd')"
        }]
    }],
    "data-points-limit": $yab.eval('${instancekey}.data-point-limit'),
    #if (${mode} == "append")
    "appendFile": "appendFile.json",
    #end
    "source-tags": {
        "namespace": "com.qad.tags",
        "linkTypes": [
            "comQadApp",
            "comQadKpi"
        ],
        "ids": [
            "!urn:app:com.qad.*",
            "!urn:kpi:com.qad.*"
         ]
    },
    "source-url": "$yab.eval('${${instancekey}.lps}.url')",
    "source-user": "$yab.eval('${${instancekey}.lps}.username')",
    "source-pwd": "$yab.eval('${${instancekey}.lps}.password')",
    "target-url": "$yab.eval('${composerinstancekey}.url')",
    "target-user": "$yab.eval('${composerinstancekey}.users.admin.username')",
    "target-pwd": "$yab.eval('${composerinstancekey}.users.admin.password')"
}
```

3. In the local copy, replace the contents of the `ids` property inside the `source-tags` property with the list of app URIs to be migrated. Do not make any other changes to the file. The following example shows a reconfigured template file with the correct syntax for the modified `source-tags` property. In this example, the URI of the app to be migrated is `urn:app:com.qad.qadextensions`. Multiple apps can also be specified by entering a comma-delimited list of URIs inside the `ids` array of this file, instead of only a single entry. In addition, an asterisk (*) character can be placed in any URI value as a wildcard to include multiple apps with a single entry (example: `''urn:app:com.thirdparty.*`).

```
{
    "connection-timeout": "$yab.eval('${instancekey}.connection-timeout')",
    "mode": "${mode}",
    "property-replacing": [{
        "namespace": "system.connections",
        "id": "$yab.eval('sourceId')",
        "properties": [{
            "path": "payload.password",
            "value": "$yab.eval('${instancekey}.connection-pwd')"
        }]
    }],
    "data-points-limit": $yab.eval('${instancekey}.data-point-limit'),
    #if (${mode} == "append")
    "appendFile": "appendFile.json",
    #end
    "source-tags": {
        "namespace": "com.qad.tags",
        "linkTypes": [
            "comQadApp",
            "comQadKpi"
        ],
        "ids": [
            "urn:app:com.qad.qadextensions"
         ]
    },
    "source-url": "$yab.eval('${${instancekey}.lps}.url')",
    "source-user": "$yab.eval('${${instancekey}.lps}.username')",
    "source-pwd": "$yab.eval('${${instancekey}.lps}.password')",
    "target-url": "$yab.eval('${composerinstancekey}.url')",
    "target-user": "$yab.eval('${composerinstancekey}.users.admin.username')",
    "target-pwd": "$yab.eval('${composerinstancekey}.users.admin.password')"
}
```

4. Rerun the migration process using the new template file through the following YAB command. The new template file will be used only in the context of this command, without permanently changing the value of the `logi-composer.default.migration-tool.configuration-template` YAB property.

```
yab -logi-composer.default.migration-tool.configuration-template:<path to cloned template> logi-composer-default-lps-migrate-mode-append
```

Example:

```
yab -logi-composer.default.migration-tool.configuration-template:/home/mfg/migration-template-app.json logi-composer-default-lps-migrate-mode-append
```

5. Review the results of the migration in the `appendResult` file and YAB log, as described in the earlier sections of this document.

Global Order Management Distribution Processing

Global Order Management Distribution Processing enables Available-To-Promise / Enterprise Materials Transfer (ATP /EMT) Visibility and EMT Tracking across domains of multiple instances of Enterprise Edition, where you have a central instance that can have visibility into all the others.

Please contact QAD Services to implement this capability in your system. *QAD Internal Link: Distributed Processing*.

TAM Conversion and Implementation

New Installation

When TAM is enabled for a particular domain within a new installation, a conversion routine is run that converts analysis codes to analysis groups. Once complete, this conversion cannot be undone. The conversion routine does the following:

- Creates a qad_wkfl record that blocks pricing from executing while the conversion is processing, which is defined as follows:

```
qad_domain = global_domain
qad_key1 = "ANALYSIS"
qad_key2 = SessionUniqueID
qad_key3 = "AP_CONV"
```

- Converts analysis codes to analysis groups
- Converts any break categories to a corresponding analysis group
- Fixes existing price lists to replace occurrences of empty attribute and order codes with "qadall--+--+--+--+–+" (all token)
- Converts analysis code links to analysis group links
- Executes an analysis group detail build process to create anx_det records for each analysis group member
- Sets adaptive pricing control flag to true
- Sets control settings to enable adaptive pricing utilizing caching (anx_det)

```
mfc_domain=global_domain
mfc_ctrl.mfc_module="SO"
mfc_ctrl.mfc_field="pic_adaptive_pricing"
mfc_ctrl.mfc_seq=460

mfc_domain=global_domain
mfc_ctrl.mfc_module="SO"
mfc_ctrl.mfc_field="pic_use_pricing_cache"
mfc_ctrl.mfc_seq=450
```

- Enables auto-regen of anx_det records when a customer or an item is changed

```
pic__qadc01="YY"
pic_cust_regen=true
pic_item_regen=true
```

- Deletes created qad_wkfl record to allow pricing to process

Upgrade

Conversion Routine

When TAM is upgraded to 3.0 for existing customers, a conversion routine is run. The routine determines if a previous analysis code conversion routine was executed in this environment. If the analysis code conversion was executed, then it does following:

- Updates browse name for dynamic analysis groups
- Converts analysis code links to analysis group links
- Sets adaptive pricing control flag to true
- Sets control settings to enable adaptive pricing utilizing caching (anx_det)

```
mfc_domain=global_domain
mfc_ctrl.mfc_module="SO"
mfc_ctrl.mfc_field="pic_adaptive_pricing"
mfc_ctrl.mfc_seq=460

mfc_domain=global_domain
mfc_ctrl.mfc_module="SO"
mfc_ctrl.mfc_field="pic_use_pricing_cache"
mfc_ctrl.mfc_seq=450
```

- Enables auto-regen of anx_det records when a customer or an item is changed

```
pic__qadc01="YY"
pic_cust_regen=true
pic_item_regen=true
```

Post-Installation Steps

Existing TAM customers who are upgrading to TAM 3.0 must complete the following steps after the installation is complete.

1. Set up NRM in TAM 3.0 for IndirectAddress.

The next three steps are done from the QAD .NET UI.

2. Run Analysis Code Detail Build for each and every domain. This needs to be done for both Item and Customer.

3. Run `convert_indaddress_tam3.p` to convert IndirectAddress data for the new IndAddressCode field.

4. Run `convert_claim.p` to convert claim data for the new Claim screen.

5. Run `convert_earneddiscount_3031.p` to convert contract earned discount data, if currently running TAM Contracts on TAM 3.0 or earlier. The conversion program was introduced in TAM 3.1 or above. If you have questions about running this program, please contact QAD Support.

QAD CRM Calendar Integration

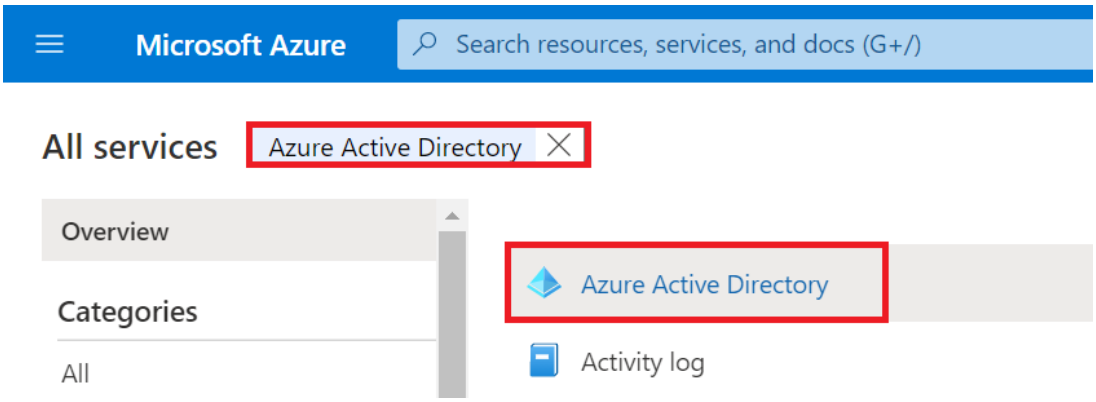
QAD CRM for Manufacturers allows you to create events that seamlessly integrate with your calendar client.

A bidirectional integration solution is presented for both Microsoft 365 Calendar and Google Calendar. For other calendar apps, a one-way directional integration solution from CRM to the calendar is available.

Setting Up Microsoft 365 Integration

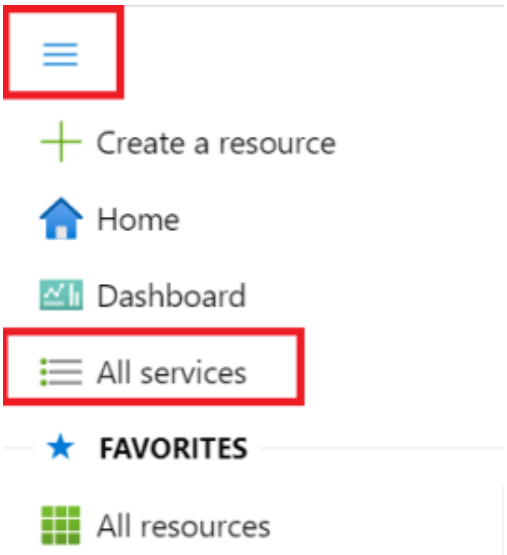
Register the QAD CRM Adaptive UX Application with Microsoft Identity Platform

1: Navigate to the Microsoft Azure Portal.

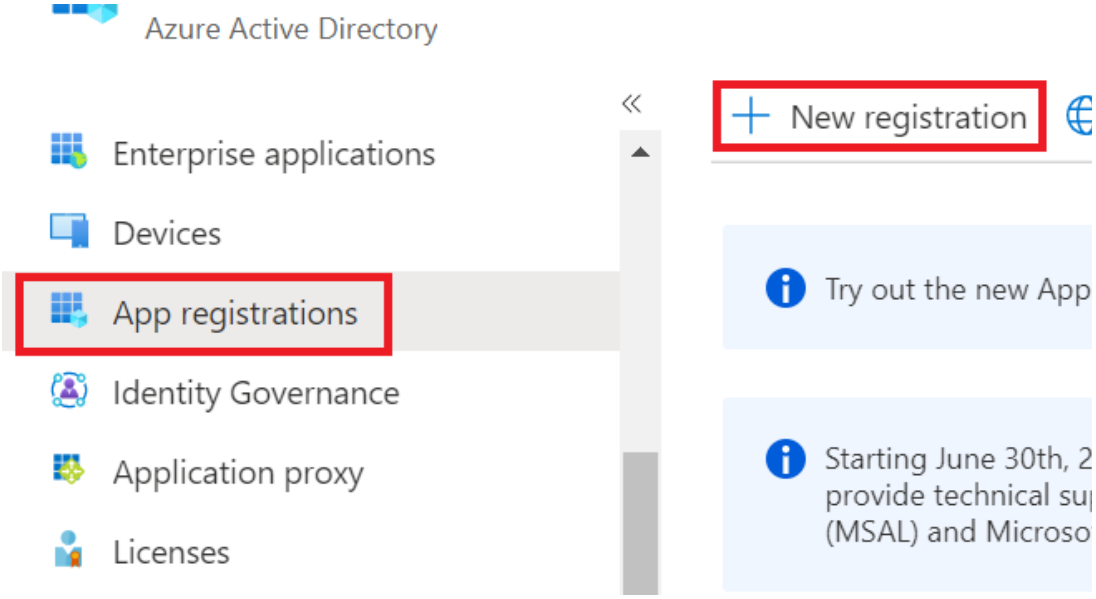


2: Create an application registration.

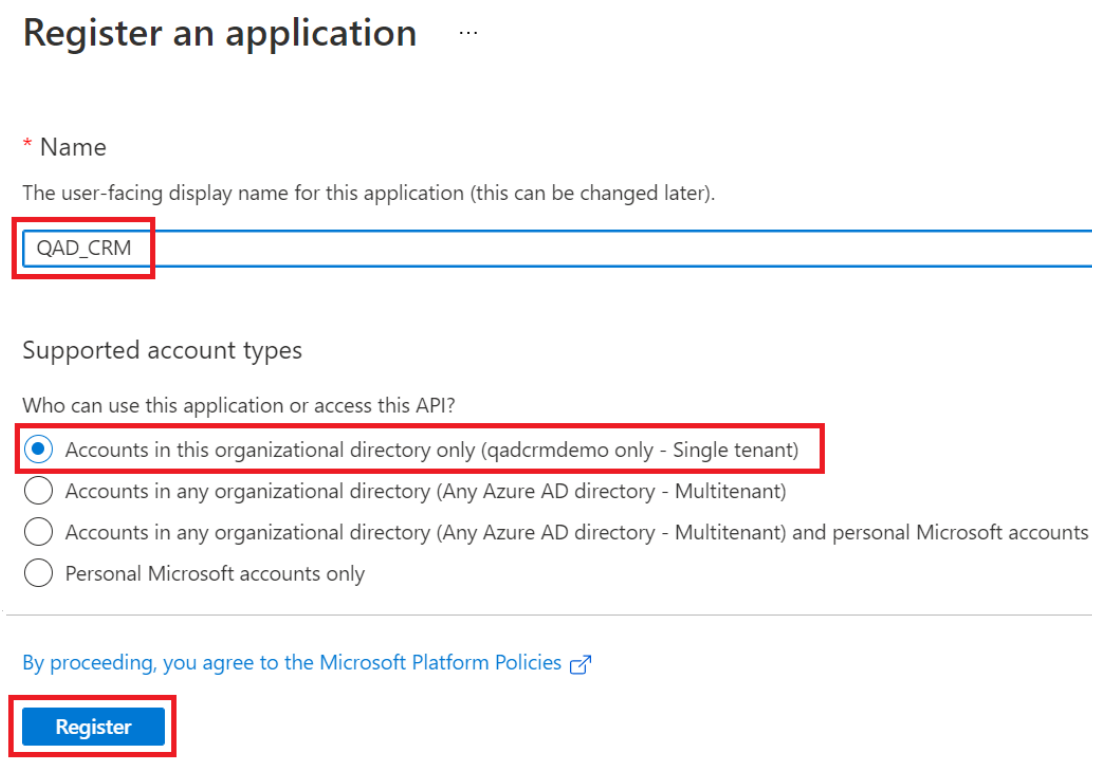
a. Open the navigation icon in the Azure Portal and select **All services.**



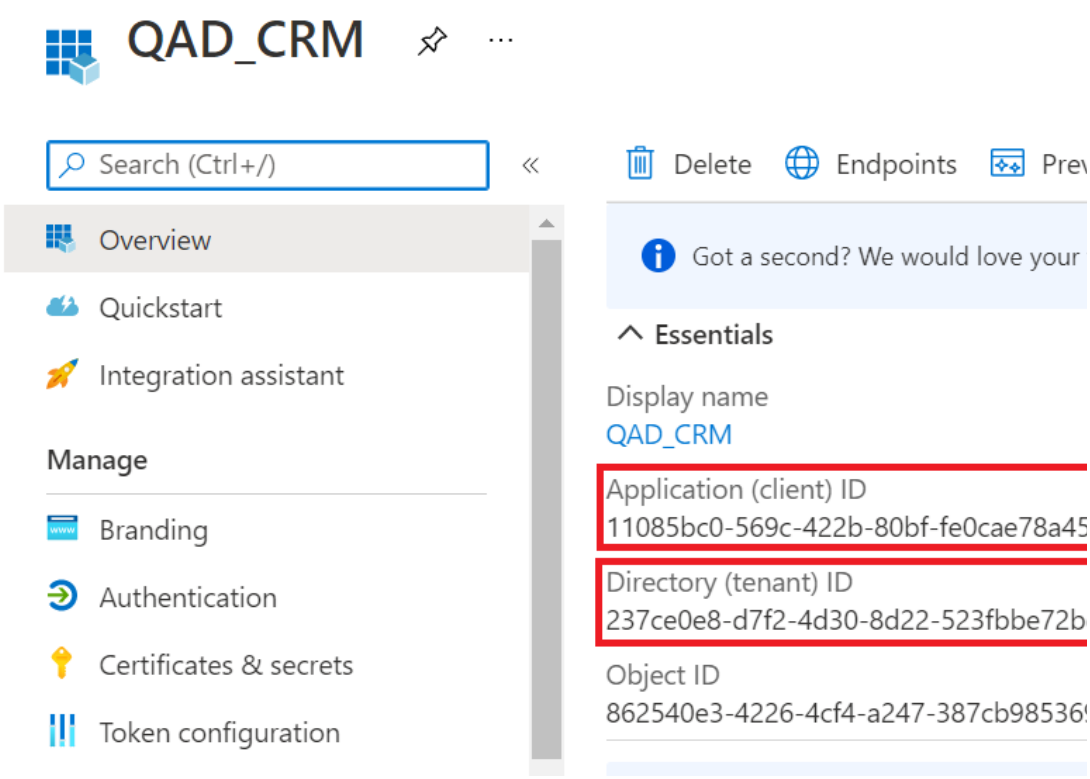
b. Select **App registrations**, then select **+ New registration.**



c. Fill in the required details as shown in the graphic and then select **Register.**

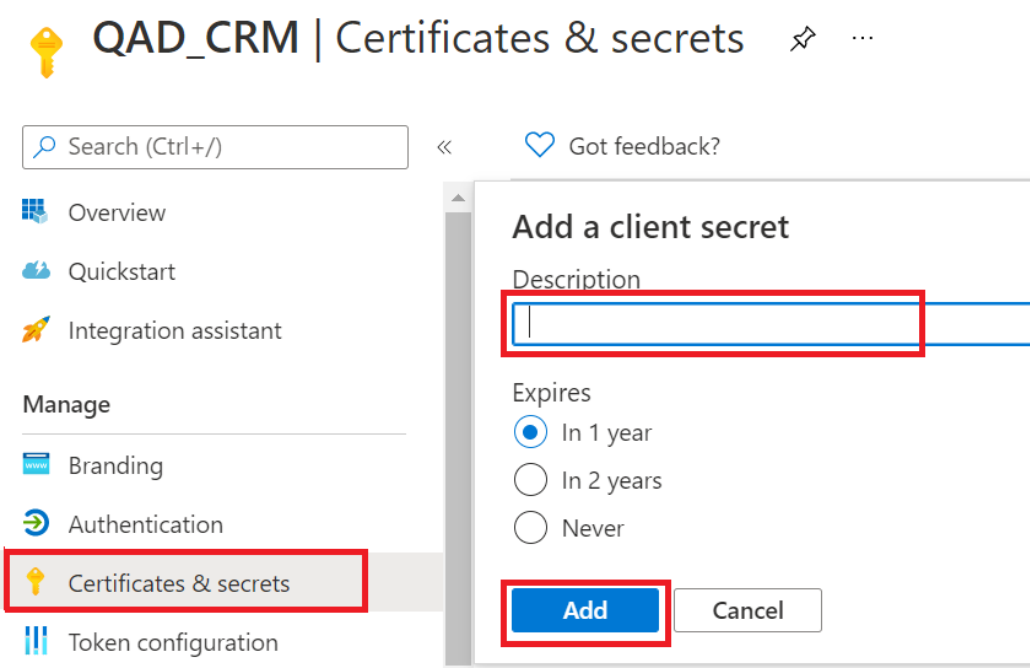


d. After registration, you must add the client ID and tenant ID to the configuration.



e. Go to the Application, select **Certificates & secrets**, then select **+New client secret.** Add a description and then select **Add.**

Note: Save or make a note of the secret as you will need to add this to the configuration.



f. Finally, navigate to the **API permissions** page. Select **Add a permission**, then select **Microsoft Graph**, then **Applicati on permissions**, then **Calendars.ReadWrite,** then **Users.Read.All**. Select **Add permissions**. Grant admin consent for your tenant.

***Note*:** API permissions must be set within the Azure portal for integration to work properly. Although you may still obtain a token, Calendar APIs within the QAD Adaptive UX may not be accessible without completing this step.

3: Update `configuration.properties` in your environment.

a. Go to `/<your env>/build/config/configuration.properties` and add the following. Replace `<client Id>`, `<client Secret>`, and `<tenant Id>` with the values you generated in Step 2, and `<public Url>` with your publicly accessible Adaptive UX home page URL.

```
# Microsoft 365 platform oauth2 configuration for Graph APIs
qad-erp-collaborationadapters.M365.oauth2.clientId=<client Id>
qad-erp-collaborationadapters.M365.oauth2.clientSecret=<client Secret>
qad-erp-collaborationadapters.M365.oauth2.tenantId= <tenant Id>
qad-erp-collaborationadapters.M365.outlook.yearsToSync=50

# Calendar push notification common settings
qad-erp-collaborationadapters.calendar.pushnotification.homepageurl=<public Url>
qad-erp-collaborationadapters.calendar.pushnotification.channelrenew.enabled=true
```

4: Run `yab update`.

Setting Up Google Calendar Integration

To enable Google Calendar Integration in Adaptive UX, a Google service account with G Suite that has granted calendar access is required to support the domain-wide delegation of authority for Google Calendar. The purpose of using Google service account is to call the Google Calendar API to maintain the calendar entries of its controlled accounts by delegation.

Creating a Service Account

A service account's credentials include a generated email address that is unique and at least one public/private key pair. If domain-wide delegation is enabled, then a client ID is also part of the service account's credentials.

1. Open the **Service accounts** page.
2. If prompted, select a project, or create a new one.
3. Click + **Create service account**.
4. Under **Service account details**, type a name, ID, and description for the service account, then click **Create**.
5. Optional: Under **Service account permissions**, select the IAM roles to grant to the service account, then click **C ontinue**.
6. Optional: Under **Grant users access to this service account**, add the users or groups that are allowed to use and manage the service account.
7. Click + **Create key**, then click **Create**.

Your new public/private key pair is generated and downloaded to your machine. It serves as the only copy of the private key. You are responsible for storing it securely. If you lose this key pair, you will need to generate a new one.

If you need to grant G Suite domain-wide authority to the service account, click the email address of the service account that you created, then copy the value from the **Unique ID** box.

To delegate authority to the service account, use the value you copied as the client ID.

Reference - https://developers.google.com/identity/protocols/oauth2/service-account

Delegating Domain-Wide Authority to the Service Account

To delegate domain-wide authority to a service account, first enable domain-wide delegation for an existing service account in the **Service accounts** page with domain-wide delegation enabled.

Then, a super administrator of the G Suite domain must complete the following steps:

1. From your G Suite domain's Admin console, go to **Main menu menu > Security > API Controls**.
2. In the **Domain wide delegation** pane, select **Manage Domain Wide Delegation**.
3. Click **Add new**.
4. In the **Client ID** field, enter the service account's **Client ID**. You can find your service account's client ID in the **Se rvice accounts** page.
5. In the **OAuth scopes (comma-delimited)** field, enter the list of scopes that your application should be granted access to. For Google Calendar Integration in Adaptive UX, it needs domain-wide full access to the Google Calendar API. Enter: https://www.googleapis.com/auth/calendar.
6. Click **Authorize**.

Your application now has the authority to make API calls as users in your domain (to "impersonate" users). When you prepare to make authorized API calls, you specify the user to impersonate.

It usually takes a few minutes for impersonation access to be granted after the client ID is added, but in some cases, it might take up to 24 hours to propagate to all users of your Google Account.

Reference - https://developers.google.com/identity/protocols/oauth2/service-account#delegatingauthority

Setting the Created Service Account in a YAB Environment

The downloaded json file of the service account's credentials should be in a format similar to the following:

json file

```
{
  "type": "service_account",
  "project_id": "qad-api",
  "private_key_id": "abcde",
  "private_key": "-----BEGIN PRIVATE KEY-----\nAAAAA\nBBBBB\nCCCCC\n-----END PRIVATE KEY-----\n",
  "client_email": "crm-google-calendar-testing@qad-api.iam.gserviceaccount.com",
  "client_id": "1234567",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/crm-google-calendar-
testing%40qad-api.iam.gserviceaccount.com"
}
```

To make the Google Calendar Integration work in an Adaptive UX environment, the credentials need to be configured in the `configuration.properties` file:

configuration.properties

```
qad-erp-collaborationadapters.calendar.google.clientid=1234567
qad-erp-collaborationadapters.calendar.google.clientemail=crm-google-calendar-testing@qad-api.iam.
gserviceaccount.com
qad-erp-collaborationadapters.calendar.google.privatekeyid=abcde
qad-erp-collaborationadapters.calendar.google.privatekey=AAAAABBBBBCCCCC
```


Remove the carriage returns in "AAAAA\nBBBBB\nCCCCC" when setting the private key in the YAB property. You do not need to copy the private key's "-----*BEGIN PRIVATE KEY*-----\n" and "\n-----*END PRIVATE KEY*-----\n" to the YAB property. As illustrated in the previous example, the final private key in the YAB property is "*AAAAABBBBBCCCCC*".

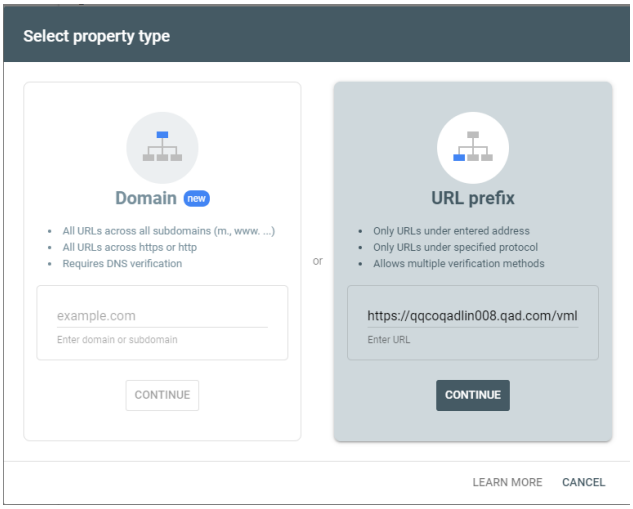
Enable Push Notifications

To eliminate the extra network and compute costs involved with polling calendar events to determine if they have changed, the Google Push Notifications allow you to watch for changes to calendar events. Whenever a watched resource changes, the Google Calendar API notifies the QAD Adaptive UX system. The following steps are required to enable push notifications.

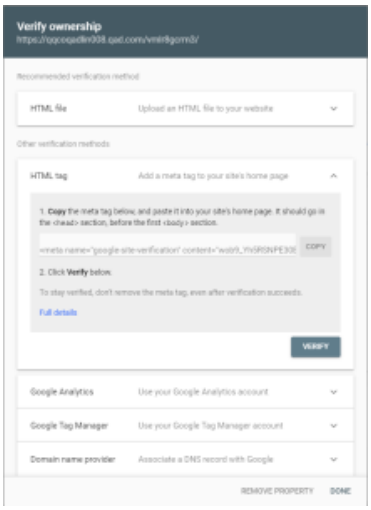
Verify that you own the domain - Before you can register your domain, you need to verify that you own it. Complete the site verification process using Search Console. For more details, see the site verification help documentation.

Details

Paste the homepage URL of the Web UI. Ensure the URL is public and accessible by Google and click "CONTINUE". In this case, https://qqcoqadlin008.qad.com/vmlr8gcrm2/ is used.



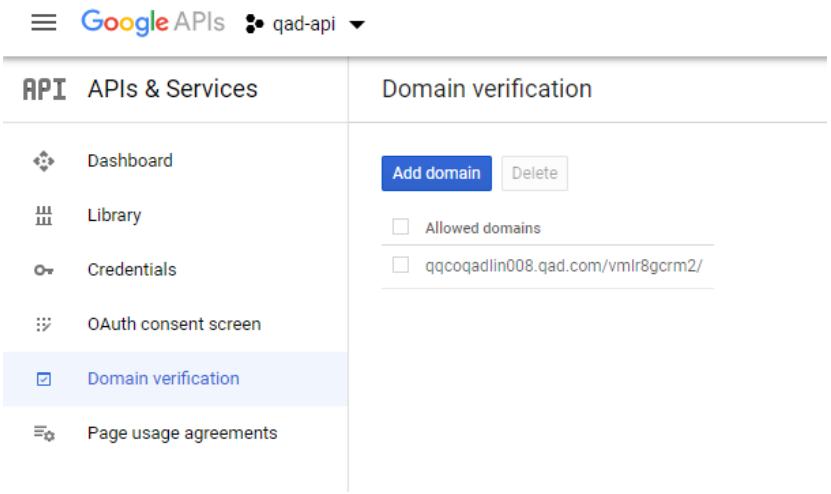
Choose the "HTML Tag" method to validate the site. Copy the tag (e.g. <meta name="google-site-verification" content="wob9_Yh5RSNPE30EmABnZhd8lq8vmNdN0F2hNRp4KI0" />)



Enter the tag content (for example: "wob9_Yh5RSNPE30EmABnZhd8lq8vmNdN0F2hNRp4KI0") in configuration. properties, and run `yab webapp-webshell-update` and `yab tomcat-webui-restart`.

```
qad-webshell.googleSiteVerificationContent=wob9_Yh5RSNPE30EmABnZhd8lq8vmNdN0F2hNRp4KI0
```

Register your domain - Go to the Domain verification page in the API Console and click **Add domain.** Fill in the form, then again click **Add domain**.



Create watch channels for users - To ask Google to send notifications to Adaptive UX, configure the following in configuration.properties to create a subscribing channel of the webhook by schedule.

configuration.properties

```
# Mandatory setting for homepage url that's defined in Google site verification
qad-erp-collaborationadapters.calendar.pushnotification.homepageurl=https://qqcoqadlin008.qad.com
/vmlr8gcrm2/
# Indicates whether periodically renew channel. Default as false, should be set to true
qad-erp-collaborationadapters.calendar.pushnotification.channelrenew.enabled=true
# How long the channels will be periodically checked for expiration. No needs to have this line
if using the default 900 seconds.
qad-erp-collaborationadapters.calendar.pushnotification.channelrenew.seconds=900
# Renew the channel for system user if the channel will be expired in certain minutes. No needs
to have this line if using the default 60 minutes.
qad-erp-collaborationadapters.calendar.pushnotification.channelrenew.minutesBeforeExpire=60
# How long the channels will be expired when creating channel. No needs to have this line if
using the default 20 day. It's allowed 24 days in maximum.
qad-erp-collaborationadapters.calendar.google.pushnotification.daysToExpire=20
```

Encrypting the Credentials with KMS

Ignore this step if KMS is already enabled.

The following settings need to be configured in the KMS along with the credentials properties above:

configuration.properties

```
kms.enabled=true
kms.server.ssl.key-store=/dr01/certificates/keystore.jks
kms.server.ssl.keyStoreType=JKS
kms.server.ssl.keyAlias=qad-wildcard
```

⚠ KMS currently does not support file encryption with YAB configuration. In a future release, the json file will be able to be placed directly into the YAB environment.

Run a yab update

After running a yab update, the client ID, private key ID and the private key will be encrypted in the `configuration.properties` file and available in the tomcat-webui, ensuring security. And for a single service account, it can be used in multiple YAB environments for same domain-wide email accounts.

Setting Up SMTP Integration

If your company does not use Microsoft 365 or Google Calendar as your email system, QAD CRM for Manufacturers provides a third option allowing you to integrate CRM events with other email client calendars.

Please be aware that this option only supports one-direction integration from CRM to the calendar. Changes made directly to the calendar cannot be integrated back into QAD CRM.

The system uses the email set below to send the notifications. Replace <email id> and <password> with the real email ID and password you want to use as the email account from which invites will be sent.

Go to /<your env>/build/config/configuration.properties and add the following:

Note: Replace <email> and <password> with the real email ID and password.

```
#CRM Calendar SMTP
qad-collaboration.email.username=<email>
qad-collaboration.email.password=<password>
qad-collaboration.email.auth=true
qad-collaboration.email.usessl=false
```

Run the command:

```
yab webapp-webshell-update tomcat-webui-restart
```

QAD CRM Email Integration

Email Integration Overview

QAD CRM for Manufacturers offers the ability to seamlessly integrate with your email clients, allowing you to send emails directly from various QAD CRM screens. The various configuration and setup steps required to use this functionality are outlined below.

Some features of email integration within the system include:

- Using an email poller program to attach email messages to proper CRM records such as CRM contacts, CRM accounts, leads, opportunities, marketing campaigns, and events.
- Sending batch emails that are handled via background processing. This function offers an email template feature where the field values of the email template can be replaced for each contact. For example, if a user wants to send a marketing email to all customers, the contact name in the template can be replaced with the contact's real name and each contact will receive their own customized email.
- A bounced email account can be configured to identify which emails are bounced. These bounced contacts can be reviewed later to correct the email ID.
- A solution to let customers unsubscribe from emails.

An Email Contacts action to send emails to multiple contacts is included for the CRM Contacts, CRM Accounts, and Lead Contacts screens as well as select other browses and grids.

Email Poller Configuration

Configuring the email poller involves setting up email addresses to:

- Act as the poller email address
- Identify bounced emails
- Unsubscribe email accounts that request it

Important: The email accounts used for all three of these purposes should not be the same account.

Go to `/<your env>/build/config/configuration.properties` and add the properties described in the following three sections. Replace `<email>` and `<password>` with the real email ID and password and replace `<protocol>`, `<hostname>`, and `<port>` with the correct information based on your email server. For example protocol=imaps, hostname=imap.gmail.com, and port=993.

Set Up CC Email Address

In order for email integration to function correctly, you need an email address to be accessed by the email poller. This email address will be CC'd on all emails sent using the system, and the poller will then copy all emails from this account allowing emails to be stored in QAD CRM.

```
qad-erp-custrelmgmt.crm.email.poller.autostart=true
qad-erp-custrelmgmt.crm.email.poller.username=<email1>
qad-erp-custrelmgmt.crm.email.poller.password=<password1>
qad-erp-custrelmgmt.crm.email.poller.protocol=<protocol>
qad-erp-custrelmgmt.crm.email.poller.hostname=<hostname>
qad-erp-custrelmgmt.crm.email.poller.port=<port>
qad-erp-custrelmgmt.crm.email.poller.mailbox=inbox
```

Set Up Bounced Emails Account

In order for the system to automatically identify bounced emails, you need an email address to be accessed by the email poller. When sending batch emails in the system using the Email Contacts action, bounced emails will be sent to this email account. The poller will then scan the bounced emails from this account and automatically mark the contact record's bounced flag as Yes in QAD CRM. It is easy to identify these invalid email addresses so that they can be reviewed and updated at a later time.

```
qad-erp-custrelmgmt.crm.email.bounced.poller.autostart=true
qad-erp-custrelmgmt.crm.email.bounced.poller.username=<email2>
qad-erp-custrelmgmt.crm.email.bounced.poller.password=<password2>
qad-erp-custrelmgmt.crm.email.bounced.poller.protocol=<protocol>
qad-erp-custrelmgmt.crm.email.bounced.poller.hostname=<hostname>
qad-erp-custrelmgmt.crm.email.bounced.poller.port=<port>
qad-erp-custrelmgmt.crm.email.bounced.poller.mailbox=inbox
```

Set Up Unsubscribe Email Account

If customers do not want to receive emails, the customers can send an email to the Unsubscribe email account. A good practice is for emails sent from the organization to include this Unsubscribe email account giving them to the option to unsubscribe. The email poller will scan the email account and mark the Email Opt Out flag of the contact as Yes in QAD CRM. When sending batch emails, the contacts with Email Opt Out can be excluded.

```
qad-erp-custrelmgmt.crm.email.unsubscribe.poller.autostart=true
qad-erp-custrelmgmt.crm.email.unsubscribe.poller.username=<email3>
qad-erp-custrelmgmt.crm.email.unsubscribe.poller.password=<password3>
qad-erp-custrelmgmt.crm.email.unsubscribe.poller.protocol=<protocol>
qad-erp-custrelmgmt.crm.email.unsubscribe.poller.hostname=<hostname>
qad-erp-custrelmgmt.crm.email.unsubscribe.poller.port=<port>
qad-erp-custrelmgmt.crm.email.unsubscribe.poller.mailbox=inbox
```

Ensure IMAP is enabled for the email accounts. If the email account is Google account, refer to this page https://support.google.com/mail/answer/7126229?hl=en to set up IMAP and also turn on "Less secure app access" on this page https://myaccount.google.com/lesssecureapps.

Once the settings are defined in configuration.properties, run the following command:

```
yab webapp-webshell-update tomcat-webui-restart
```

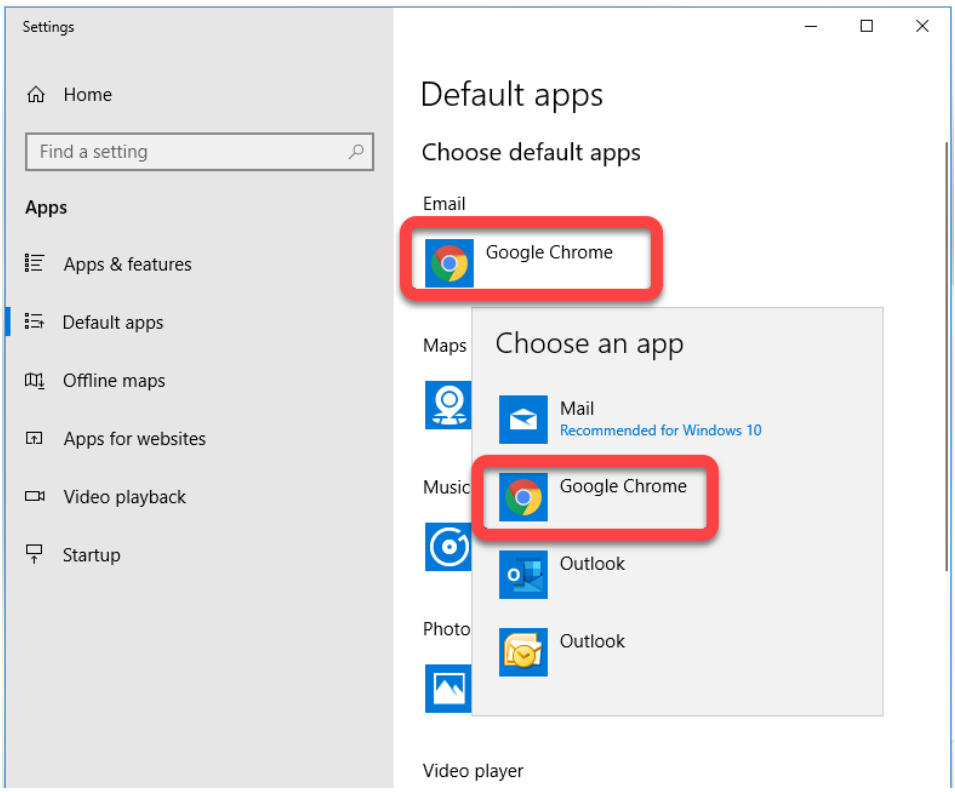
Setting a User's Default Email App

QAD CRM Adaptive UX supports email integration with various email clients. When sending an email from CRM, the system will open the email client directly. To have this process open a particular email client, users first need to set up their default email app. This section explains the steps to perform on end user devices to enable the functionality after the email poller configuration is complete. Your end users may require assistance in completing these steps.

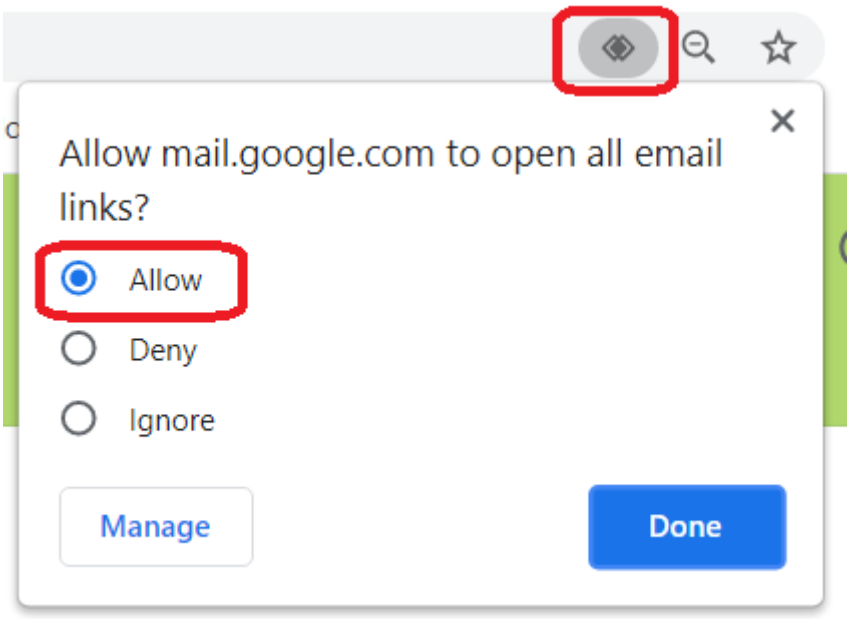
Use the instructions below for your email app of choice.

Setting Gmail as the Default Email App on Windows

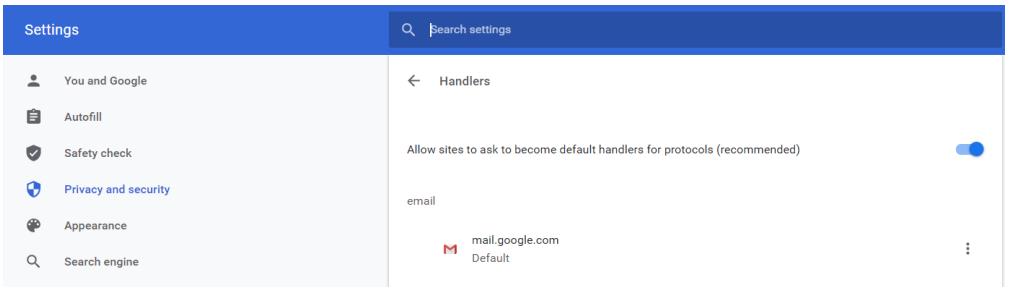
For users who normally use Gmail, open Windows Settings and go to Default apps. Select Google Chrome as the default app.



Then, go to chrome://settings/handlers and check if mail.google.com is present; if it is, set it as default. If it is not, reload Gmail and click on the icon to the left of the search icon, as seen in the image below. Select Allow and click Done.

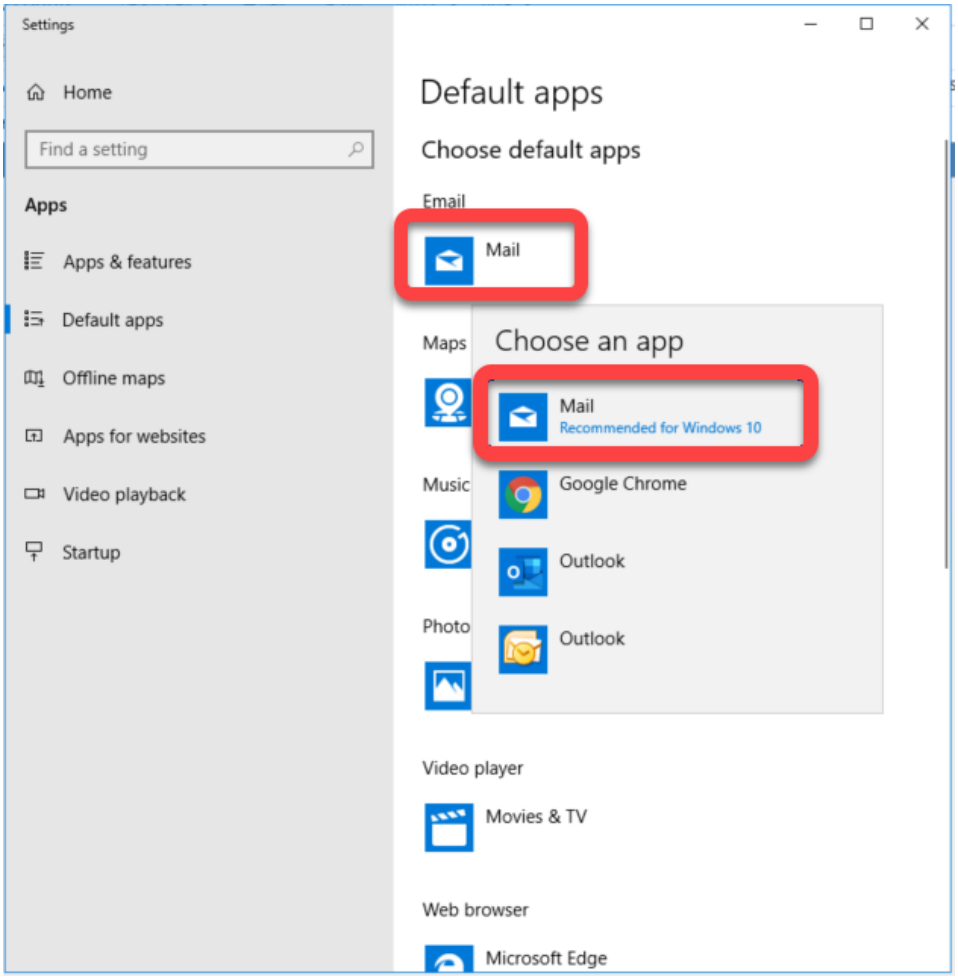


To find Handlers, go to the Privacy and security section. Then go to Site Settings and click Additional Permissions.



Setting Mail as the Default Email App

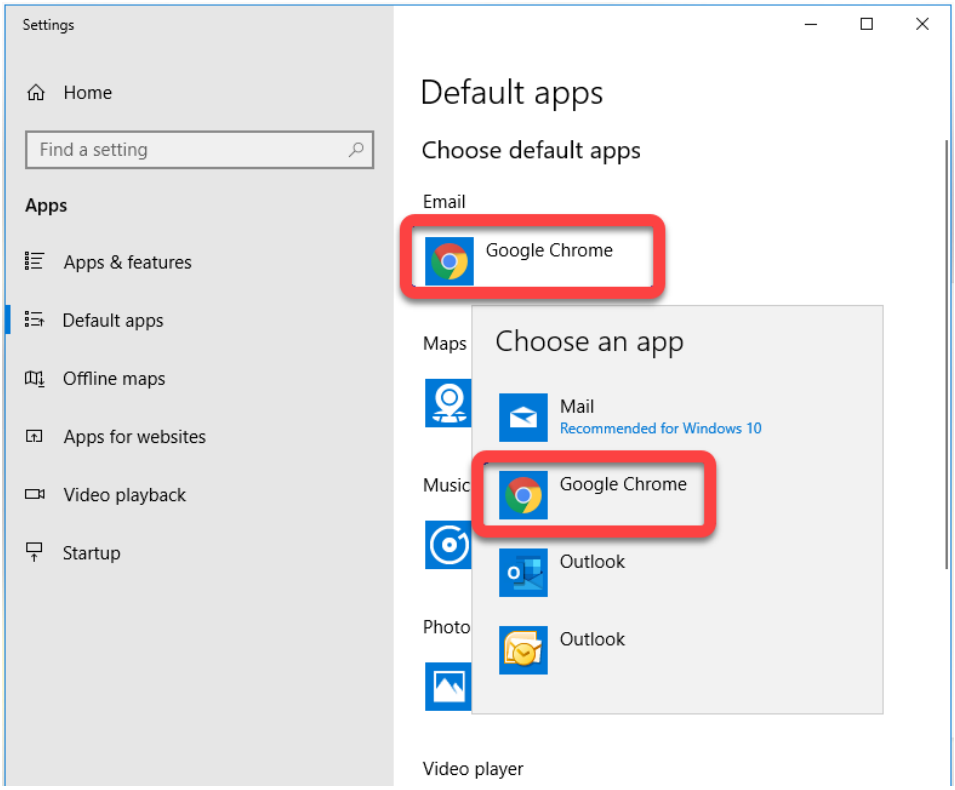
In Windows 10, Mail is set as the default email app. If a user normally uses this app, then no extra setup is required.



Setting Outlook 365 Web Client as the Default Email App

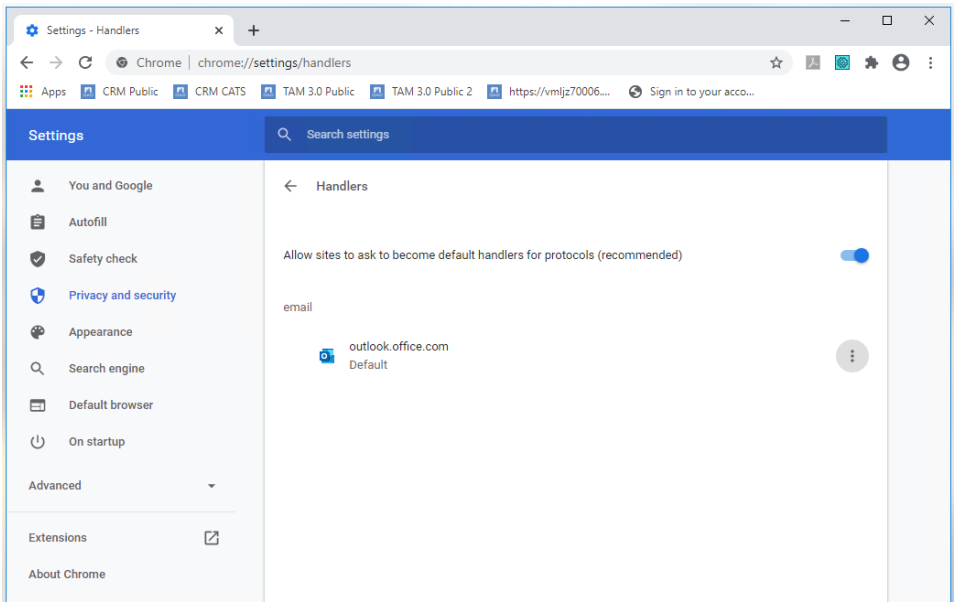
For users who normally use Outlook 365 web client, open Windows Settings and go to Default apps.

Select Google Chrome as the default app.



Installing Outlook Handler in Google Chrome

Go to Chrome://settings/handlers and check if Outlook is present. If so, set it as the default.



If Outlook is not present in Chrome://settings/handlers, log into Outlook 365 web client in Chrome, press F12 to open DevTools.

Enter the following command in the console:

```
navigator.registerProtocolHandler("mailto","<outlook 365 website>/owa/?path=/mail/action
/compose&to=%s","Outlook")
```

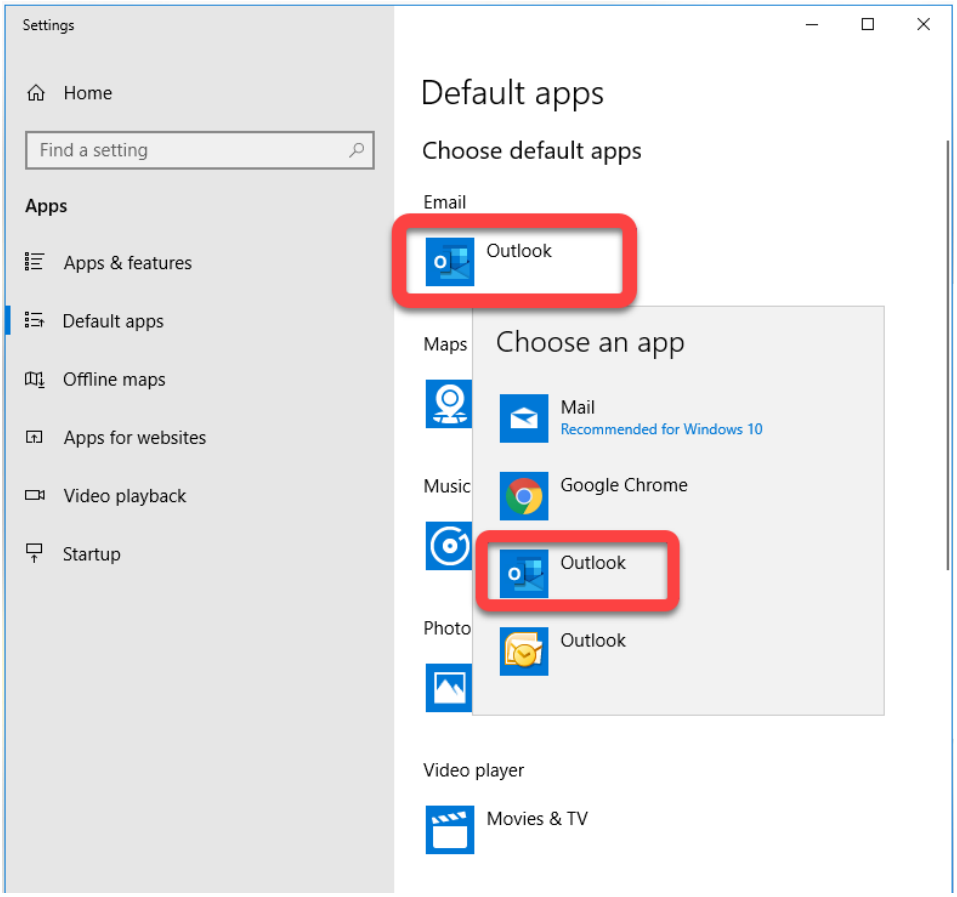
Replace `<outlook 365 website>` with the one you are currently using and press Enter to save.

For example:

> *If the user logs into "https://outlook.office.com/mail/inbox", then the command used should be:*
> *navigator.registerProtocolHandler("mailto","https://outlook.office.com/owa/?path=/mail/action/compose&to=%s","Outlook")*

Setting Outlook 365 Windows Client as the Default Email App

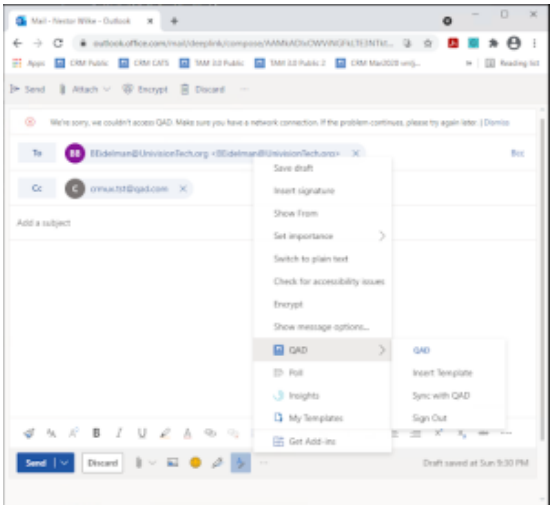
Once the Outlook 365 Windows app is downloaded, open the Windows Settings and go to Default apps. Select Outlook as the default app.

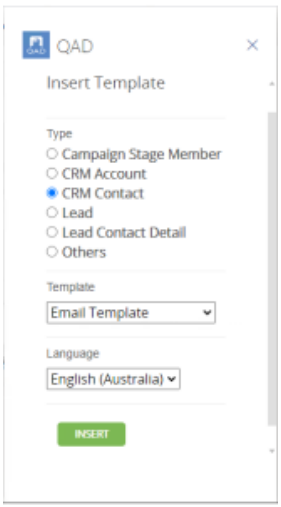


Then go to Chrome://settings/handlers and remove all email handlers.

Installing the QAD Add-on in Outlook

Email templates can be maintained using Notification Templates in QAD CRM. Using a QAD add-on, users can insert a predefined email template onto the Outlook email composition screen. This optional add-on also can be used to sync the email history back to CRM.





The user's email history is synced to QAD CRM using the email account previously set up as the Cc email address in the **Set Up CC Email Address** section. This address must be entered in the To, Cc or Bcc field.

Please follow the steps below to install the QAD Add-on for Outlook.

Generate the Outlook Add-on Manifest File

Open the URL https://<aux host>/clouderp/api/open/custrelmgmt/manifest.xml in a web browser.

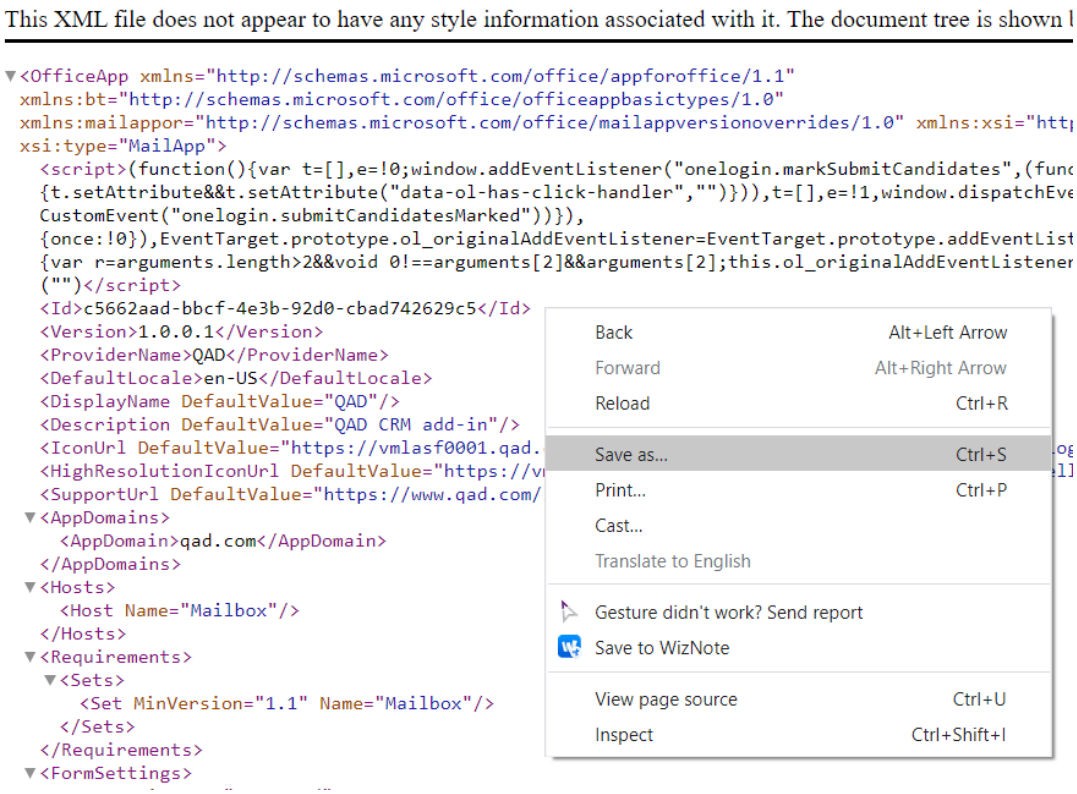
For example:

> *https://vmlasf0001.qad.com/clouderp/api/open/custrelmgmt/manifest.xml*

Save the result page as an xml file.

For example:

> *manifest.xml.*



Install the Manifest File in an Outlook Web Client

Follow the instructions here: https://docs.microsoft.com/en-us/office/dev/add-ins/testing/sideload-office-add-ins-for-testing

Install the Manifest File in an Outlook Windows Client

Follow the instructions here: https://docs.microsoft.com/en-us/office/dev/add-ins/testing/create-a-network-shared-folder-catalog-for-task-pane-and-content-add-ins

Implementation FAQ

A successful implementation requires you to gather and understand knowledge and data from a wide range of QAD resources. This section addresses common questions about installation and implementation and when applicable, points you to the source documents for the answers.

QAD Sites

Use the following links to access training, documentation, partner information, and more:

- QAD Certification Guide
- QAD Forums
- QAD Document Library
- QAD Support Site
- QAD Partner Portal
- QAD Services Partner Network (QPN)
    - Sizing and Tuning Guides
- QAD Learning Management System (LMS)

FAQ Topics

- Operation System Sizing
- Operating System Configuration
- Installing Adaptive UX

Operation System Sizing

For the most up-to-date information on sizing and tuning requirements for different releases of QAD Adaptive ERP, visit the Sizing and Tuning Guides section of QPN. The tables on this page are taken directly from the *QAD Adaptive ERP Sizing Guide*.

What is the Amount of Memory Required to Run One Environment?

Recommended sizing is as follows for production deployments.

Memory Sizing

All values are in gigabytes (GB)

User Count	< 50	< 100	< 200	< 400	< 800	< 1000	< 2000
QAD Adaptive UX	32	48	64	112	144	200	248
QAD Enterprise Edition	16	16	24	48	96	104	168

NOTE: Figures do not include sizing for any other applications, such as monitors, logstash, anti-virus, and icinga.

What are Basic Guidelines to Prepare a Sizing for a Customer?

CPU Sizing

QAD Version	Minimum CPUs	First 1000 users	Additional Users
QAD Adaptive UX	4	25 users per core	30 users per core
QAD Enterprise Edition	2	40 users per core	60 users per core

Note 1: If sizing is less than the minimum number of CPUs, use the minimum number.

Note 2: Figures do not include sizing for any other applications, such as monitors, logstash, anti-virus, and icinga.

Note 3:

- Very heavy concurrent Action Center use **may require additional CPU Resources.**
- Activity Feed Entity tracking for calculated fields results in additional API fetches, which **may require additional CPU resources.**
- Bulk record changes for fields that have Activity Tracking enabled for non-database fields **may require more CPU resources** to avoid the server becoming overloaded or Activity Feed events being delayed.

Assumptions:

- Moderate Action Center usage
- Moderate Entity Field Tracking on calculated fields (non-database fields)

Operating System Configuration

```
vm.swappiness = 10
vm.max_map_count = 262144
kernel.shmmax = <RAM SIZE>
kernel.shmall = <SHMMAX>
kernel.shmmni = 4096
kernel.sem = 10000 640000 2560 20480
fs.file-max = 100000
net.ipv4.ip_local_port_range = 1024-14999,30001-65000
net.ipv4.ip_local_reserved_ports = 15000-30000
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
```

What Kernel Settings Are Required?

Change the first line, vm.swappiness, as show in bold in the following example.

vm.swappiness = 1
vm.max_map_count = 262144
kernel.shmmax = <RAM SIZE>
kernel.shmall = <SHMMAX>
kernel.shmmni = 4096
kernel.sem = 10000 640000 2560 20480
fs.file-max = 100000
net.ipv4.ip_local_port_range = 1024-14999,30001-65000
net.ipv4.ip_local_reserved_ports = 15000-30000
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1

How Many Semaphores Are Required Per Environment?

These are set by the values in kernel.sem. Generally speaking, Progress needs two semaphores per active database, per this entry in the Progress Knowledge Base.

The knowledge base entry includes the following details. The total number of required semaphores is dependent upon:

- The number of concurrent users (add one semaphore per 50 users)
- Before Image Buffers (add one semaphore per 400 -B)
- The number of active databases on the system
- The number of semaphores the OS uses (kernel, daemons etc.)
- The number of any third-party applications / services / daemons etc. that are being used.

The semaphore usage is presented per Progress guidelines. Manual tuning of these parameters may be necessary. They recommend testing and monitoring in order to identify the optimal value for the system.

Installing Adaptive UX

What is the Web UI Proxy URL?

During Adaptive UX installation, the installer prompts you to enter an optional Web UI Proxy URL. This is typically an Apache server instance used to control external access to the Web UI.

The Apache Reverse Proxy server sits in the DMZ, accessible to the public internet, and serves as a gateway to Tomcat servers running the Web UI. Because it is publicly accessible, the Apache server must be protected with basic security hardening measures, including the use of SSL/TLS for all communication. The server's main responsibilities are to pass HTTPS requests to the correct Tomcat server and to enable compression for performance reasons. Compression is critical if the Web UI will be accessed over a WAN.

The Apache Reverse Proxy should be set up to support:

1. Configuration for Action Centers / Logi. Refer to the Action Centers section of this guide and the *QAD Security Administration Guide* (September 2020 version) on the QAD Document Library.
2. Compression. The information is outlined below as a guide, but refer to the official Apache documentation.

Load filter and deflate modules:

/etc/httpd/conf/httpd.conf

```
LoadModule filter_module modules/mod_filter.so
LoadModule deflate_module modules/mod_deflate.so
```

Define filters:

/etc/httpd/conf.d/filter.conf

```
#Output filter definitions
# Inflate filter for compressed content from remote servers
FilterProvider gzinflate INFLATE resp=Content-Encoding $gzip

# Substitute filters for fixing links
FilterProvider subst SUBSTITUTE resp=Content-Type $text/html
FilterProvider subst SUBSTITUTE resp=Content-Type $application/json

# Deflate filter for requests that accept gzip encoding
FilterProvider gzdeflate DEFLATE req=Accept-Encoding $gzip
```

Using filters in proxy configuration

```
<Location /compression>
        SubstituteMaxLineLength 10m
        ProxyPass https://tomcat.qad.com:34011/qad-central
        ProxyPassReverse https://tomcat.qad.com:34011/qad-central
        ProxyPassReverseCookiePath /qad-central /compression
        Header edit Location "/qad-central/" "https://dmz.qad.com/compression/"
        FilterChain +gzinflate +subst +gzdeflate
        Substitute "s|/qad-central|/compression|iq"
</Location>
<IfModule mod_deflate.c>
        SetOutputFilter DEFLATE
        Header append Vary User-Agent env=!dont-vary
        DeflateCompressionLevel 9
        AddOutputFilterByType DEFLATE text/plain
        AddOutputFilterByType DEFLATE text/html
        AddOutputFilterByType DEFLATE text/xml
        AddOutputFilterByType DEFLATE text/css
        AddOutputFilterByType DEFLATE application/xml
        AddOutputFilterByType DEFLATE application/xhtml+xml
        AddOutputFilterByType DEFLATE application/rss+xml
        AddOutputFilterByType DEFLATE application/javascript
        AddOutputFilterByType DEFLATE application/x-javascript
        AddOutputFilterByType DEFLATE application/json
        AddOutputFilterByType DEFLATE image/svg+xml
        AddOutputFilterByType DEFLATE text/javascript
</IfModule>
```

What is the Impact of Choosing a Customer Vertical?

The customer vertical information is used within the Guide Me capability and allows the on-screen help to be tailored to the customer's specific vertical.

What is the Impact of Choosing an Environment Type?

There are a number of places where the environment type is used to enable or disable certain capabilities in the application.

- Certain platform development features are only enabled in a Development environment and not in a Test or Production environment.
- The licensing of some products can differ based on whether the environment is a Production environment or not.
- UI theming can be based on the environment type, so different color schemes can be used for a Dev, Test or Prod environment.

What is the Minimum Port Range Required Per Environment?

This can vary greatly depending on the products installed and how the environment is tuned (i.e. the maximum number of agents defined for each appserver) but the standard default port range for statically defined ports is 200 (22000-22200).