QAD Enterprise Applications
Channel Islands 2019

Implementation Guide

QAD Channel Islands On-Premise

QAD Enterprise Applications

Channel Islands 2019

March 2019

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

Table of Contents

Channel Islands On-Premise Implementation Guide

This guide provides technical information for implementing various features of the Channel Islands 2019 release into an existing QAD Enterprise Edition environment.

Before proceeding with any installation or implementation tasks, be sure to read the *QAD Cloud ERP with Channel Islands 2019 Release Notes*.

Channel Islands Installation

For QAD Cloud ERP with Channel Islands, QAD performs and manages the installation. For on-premise installations of Channel Islands 2019, see the *Channel Islands 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

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.

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.

Action Center Technical Overview

The Technical Overview section describes the high-level technical architecture behind the Action Centers. The overview introduces the most important components of the technical architecture, which changed significantly with the September 2018 release of Channel Islands.

Key Concepts

Reasons for Change

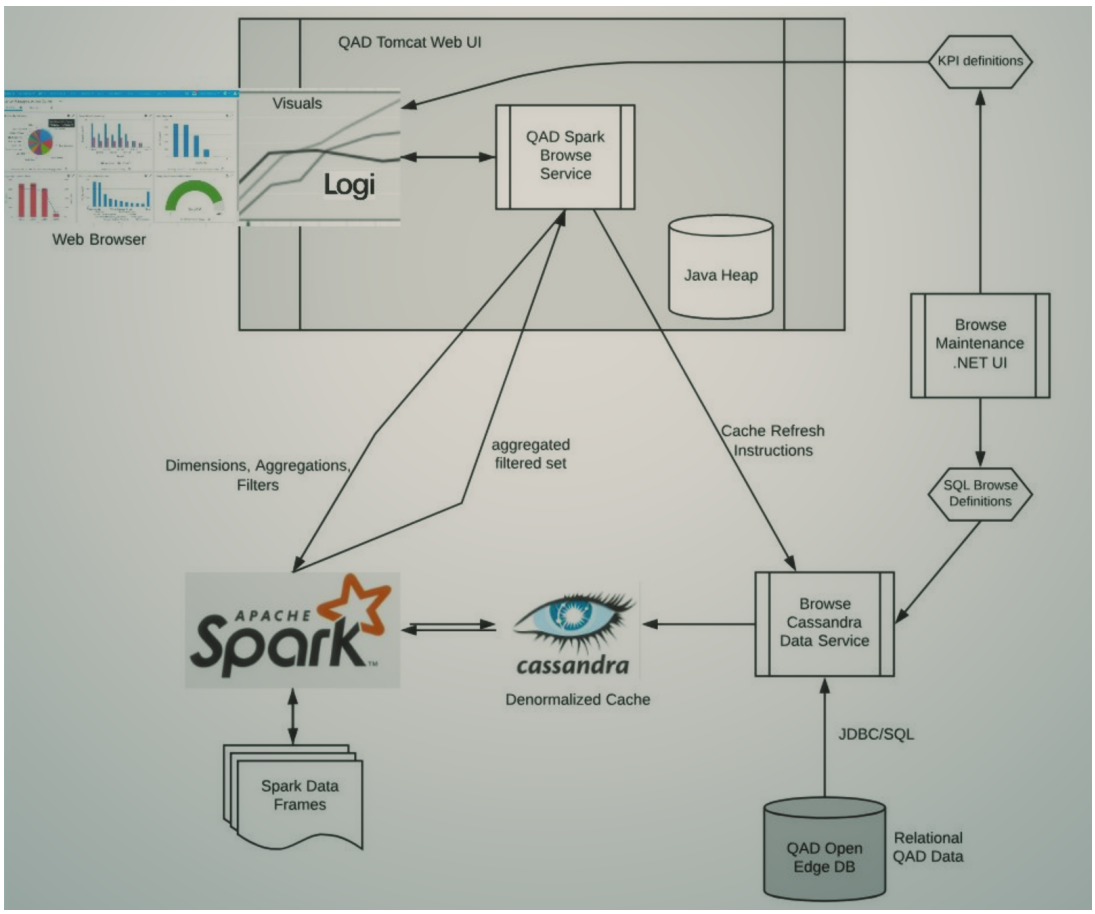
Previous Action Center releases had performance limitations when summarizing and displaying large amounts of source data, principally the very large result sets returned by some browses. In particular, performance degraded rapidly as the number of records returned by a browse increased. To mitigate this 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 new 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

New infrastructure is provided to process many existing browses using SQL retrieving 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.

In the current release, SQL-based data retrieval is supported for a subset of browses, while other browses continue to rely on the older AppServer-based browse engine. In future releases, new browse metadata and development tools will be introduced in order to migrate more existing browses to 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.

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 very 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 run time when an Action Center is displayed, the data is pre-grouped and pre-aggregated by Spark based on inputs received from the Action Center, as described in the next section. 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

Previous 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 webapp 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 webapps are used to communicate user directives, metadata, and business data into the Logi environment.

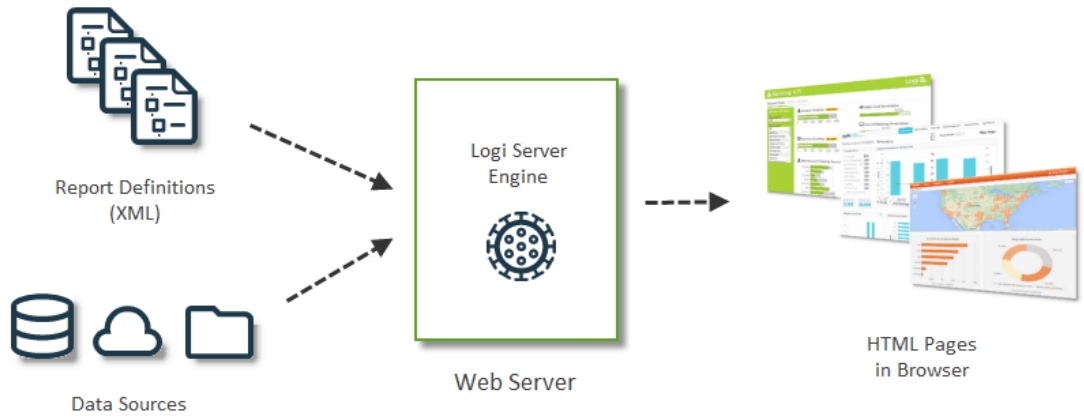
Beginning with the September 2018 release, the Logi integration was enhanced to take advantage of the superior performance of the new Query Service. As end users design visuals and their data contents using the Action Center's self-service capabilities, metadata defining the group-by categories and numeric 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 amount 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.

Action Center Key Components

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

Logi Info

Logi Info, a product from Logi Analytics, is a framework for developing and displaying analytical data embedded inside a host application, in this case QAD Enterprise 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.

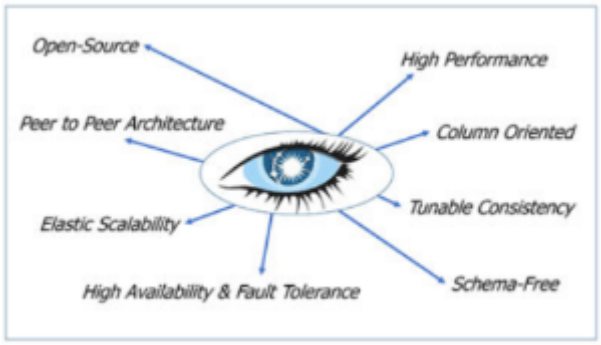
1. *Report Definitions* are text files that contain the information describing report layout and contents, stored as XML documents. While it is possible to create and edit definitions with any text editor, Logi Studio provides an integrated development environment with tools and helpful wizards that 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.

Logi Info has been used in previous Action Center releases, but its self-service capabilities are enhanced to consume larger volumes of source data in a more performant way using the new Query Service's other components, which are described in the following sections.

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 cl uster 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 dropped and re-created. 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.

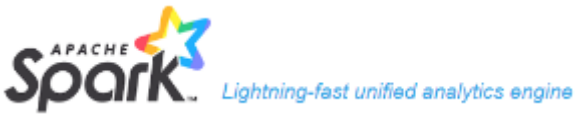
Full consistency has a negative effect on cost-effective horizontal scaling. If the database needs to check the consistency of every transaction continuously, a database with billions of transactions will incur a significant cost to perform all the checks. The idea of consistency is not practical in 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. Joins 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. Tombstones also consume disk storage unnecessarily.

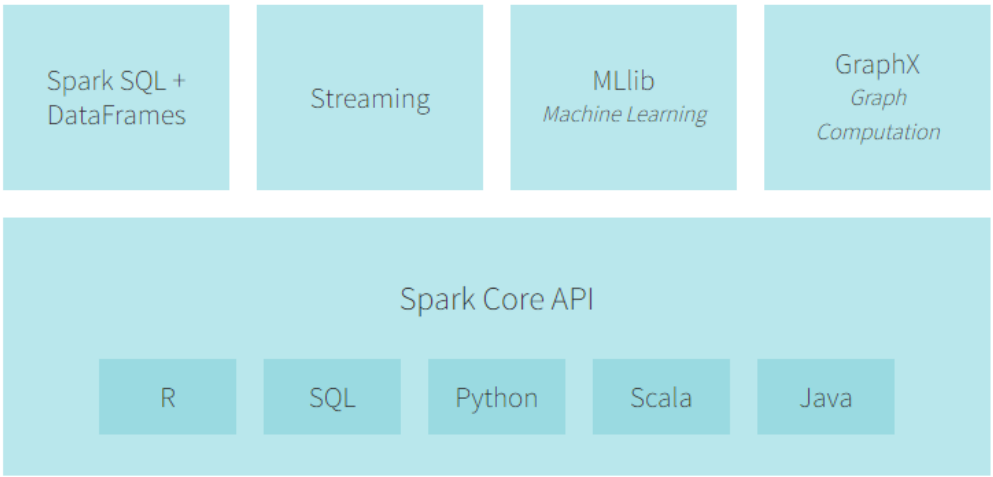
Apache Spark



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.

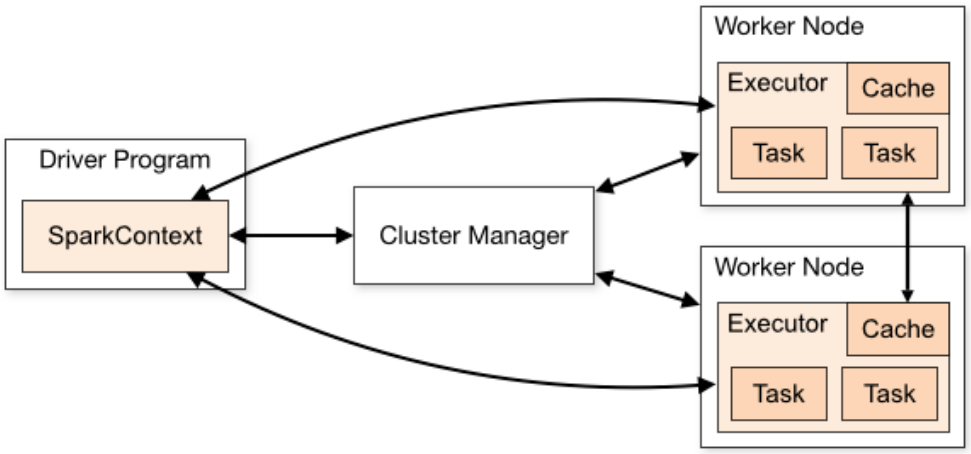


The current Query Services uses Spark's native Java API to read and manipulate the browse data.

Architecture

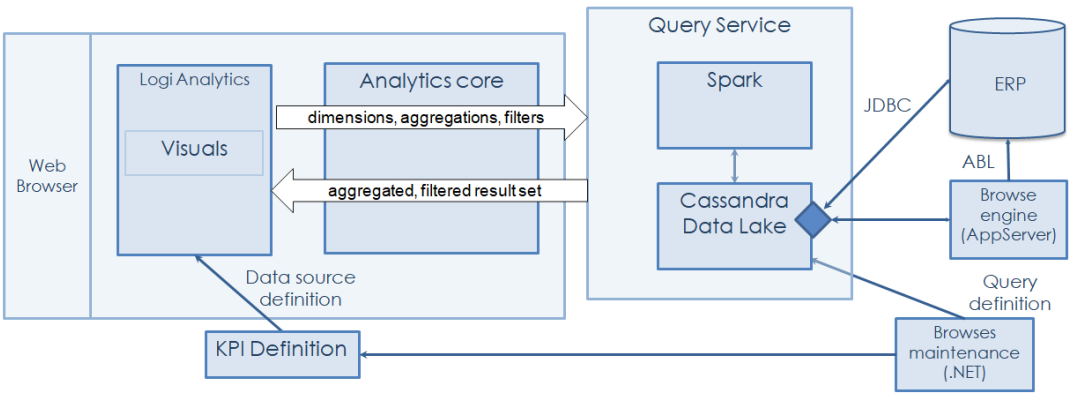
Spark applications run as independent sets of processes on a cluster, coordinated by the SparkContext object in a main program called the "driver" program. The Query Service has its own driver program and SparkContext instance, from which all the in-memory browse data can be accessed.

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.



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: Browses for which SQL access is supported 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 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 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 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.

Action Center Installation

The Installation section describes useful details about how Action Centers and the related Query Service infrastructure 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 *Channel Islands On-Premise Installation Guide*. However, this section describes some installation steps in greater depth that are specific to Action Centers, referring to other guides as needed for context and YAB command details.

This document assumes that the reader is familiar with basic Linux system administration and YAB. For more details on YAB, see the *QAD Configuration and Administration Guide* for YAB 1.8.

System Requirements

Memory

Spark and Cassandra are fast because they do considerable amounts of in-memory processing. Logi Info also requires 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. See the *Channel Islands On-Premise Installation Guide* for other software prerequisites.

Installing Action Centers

Many 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 current release, the following packages include Action Centers.

- Financials
- Sales
- Purchasing
- Inventory
- Pushproduction
- Assetmgmt

These Action Centers install automatically with their respective packages as part of the YAB metadata-update command and its sub-commands. To install the metadata for a single package, including any 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*.

Action Centers and KPIs created in custom packages are installed in the same way as described for pre-defined Action Centers.

Action Centers Are Installed But Not Updated

There is an important difference in the handling of Action Centers vs other kinds of metadata: Action Centers 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 existing 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 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.

Logi License

Logi Info is a proprietary product that requires a license file obtained from QAD. To use Action Centers, you will need to apply the Logi license file you received from QAD.

This file should be copied into the root directory of the qad-dashboards webapp. To find this root directory run the following command.

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

Exporting Action Centers

When Action Centers and KPIs are created by users, they are automatically associated with the default app for the environment. Those 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.

- 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

To obtain a list and description of all YAB commands that can be used to deploy and extract Logi Action Center files, 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
```

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
```

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*.

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. For more information on Action Center permissions, see the online help for the QAD Web UI (https://documentlibrary.qad.com/help/webui/sanmiguel/en-US/index.html#page/QAD%2520Online%2520Help%2FActionCenterPermissions.html).

Domain-Entity Membership

Another aspect of Action Center security is the membership of users within particular domains and financials entities. This type of security is part of the common Web UI security infrastructure, and is not covered in this document. However, domain and entity membership is 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 Info Authentication

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

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._base.user
cassandra._base.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-qracore.spark.ui.port
```

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

```
qad-qracore.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-qracore.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._master.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._master.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._slave.env.webui.port
```

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

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.

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)

The infrastructure used to cache the data for these types of KPIs are different, but most of the caching configuration is defined using properties common to all KPI types.

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 tables 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;

mfg_wo251_10usa                     mfg_po042_20fra
mfg_ea023                           kpi_sales_analysis_2_21nlco
mfg_ic763_10usa                     mfg_po042_11can
mfg_po041_10usa                     kpi_sales_analysis_2_10usaco
mfg_po042_31aus                     mfg_ic752_10usa
mfg_po042_21nl                      mfg_gp823
mfg_wo254_10usa                     mfg_wo253_10usa
browsecopystatus                    kpi_sales_analysis_2_40brzco
mfg_ic763_11can                     mfg_wo257_10usa
kpi_sales_analysis_2_22ukco         mfg_wo252_10usa
kpi_sales_analysis_2_31ausco        kpi_sales_analysis_2_20fraco
mfg_po042_10usa                     mfg_ic763_12mex
```

```
kpi_sales_analysis_2_21itco           kpi_sales_analysis_2_31apcons
mfg_po042_30chn                       kpi_sales_analysis_2_12mexco
mfg_po042_12mex                       mfg_an001_10usaco
fin_bdinvoice_selectdinvoice_10usaco  kpi_sales_analysis_2_30chnco
mfg_wo258_10usa                       kpi_sales_analysis_2_11canco
kpi_sales_analysis_2_23gerco

cqlsh:browses>
```

For example, the table 'mfg_wo251_10usa' contains the data returned by the operational ('mfg') browse 'OEE By Site - 3 Months' (urn:browse:mfg:wo251) for the domain 10USA. The table 'fin_bdinvoice_selectdinvoice_10usaco' contains the data returned by the financial ('fin') browse 'Customer Invoices' (urn:browse:fin:BDInvoice. SelectDInvoice) for the entity 10USACO. Table names beginning with 'kpi_', as well as the table 'browsecopystatus', are special cases not covered in this document.

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.

Financial Report Writer KPIs

The Financial Report Writer (FRW) is the primary financial reporting tool within the Financials application. It is maintained in the QAD .NET UI and has the capability to define financial KPIs based on the chart of accounts, organizational, and reporting structures specific to the enterprise. These FRW KPIs can organize and roll up financial measures in a manner consistent with the enterprise's financial reports, which could not be easily done through user-defined browses.

> FRW-based KPIs can only show data in action centers if FRW reports have been set up in the QAD .NET UI and FRW-based KPIs are created in the QAD .NET UI with names that match the FRW-based KPI definitions. For information on how to set up FRW reports and create FRW-based KPIs in the QAD .NET UI, see *QAD Financials User Guide*.

FRW KPIs generally consist of smaller data sets than browse-based KPIs, and are not cached using the Query Service. Instead, they are cached in memory within the product's primary web application using the Java Ehcache library.

Entries in the FRW KPI cache are identified by a combination of KPI code and financial entity, as in the following examples.

KPI Code	Entity Code
Cash Flow Analysis	10USACO
Days Payable Outstanding	21NLCO
Net Profit Margin	22EMEACONS
Working Capital	30CHNCO

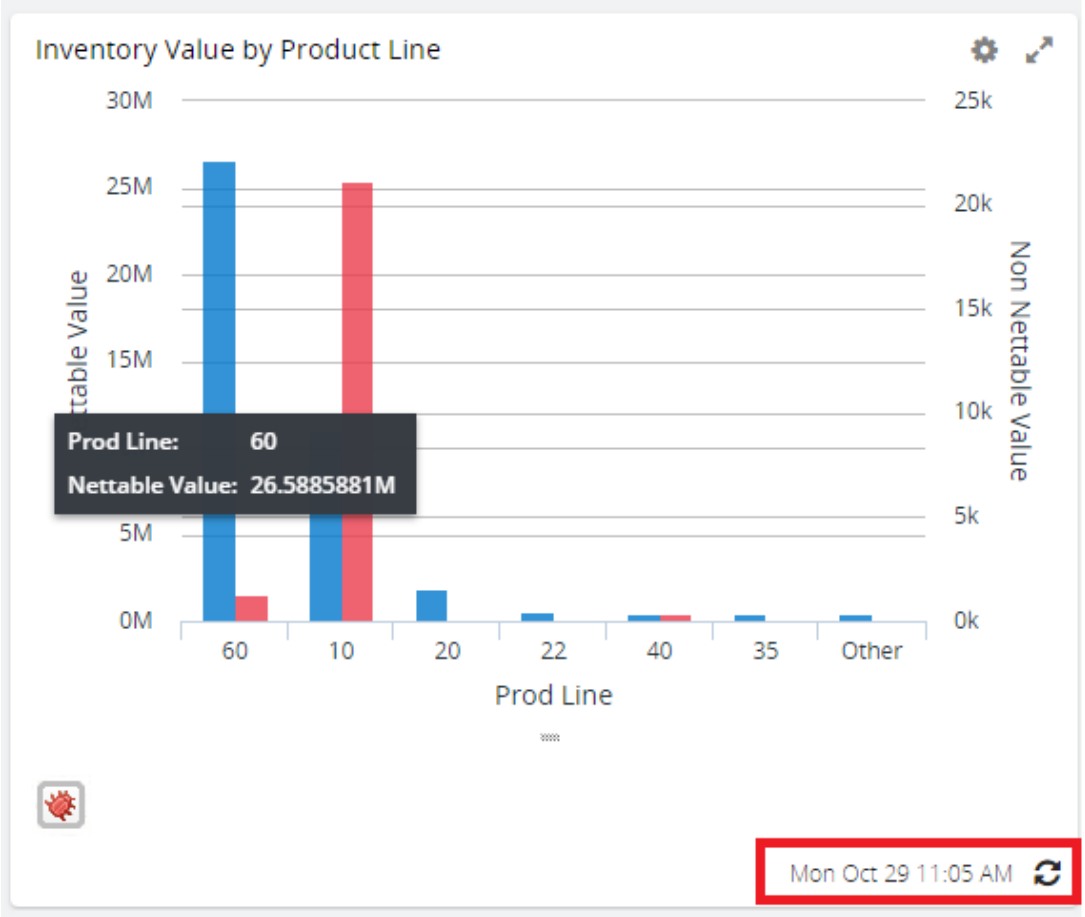
KPI Cache Refreshes

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

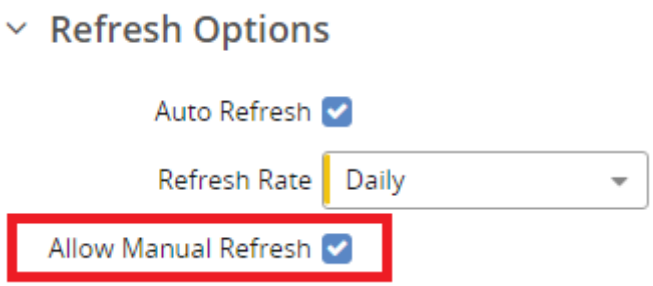
- Manual Refresh
- Cache Warming
- Scheduled Refresh

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 or FRW KPI 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 Logi Info.

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*.

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).

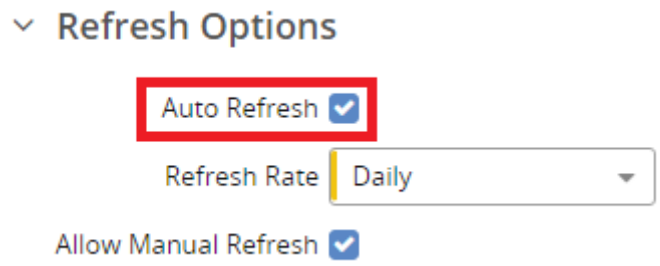
FRW KPI Cache Warming Properties

Property	Purpose/Description	Default Value	Tuning Considerations
qad-analytics-financials.cache.kpis-frw.loadAtStartup	Indicates if the cache is warmed at application startup.	true	Set to false in order to disable cache warming.
qad-analytics-financials.frwKpiCacheWarmerExecutor.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.

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	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 on-line user activity.

	the schedule for background refresh activity run within the Tomcat environment.		
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 2 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 tha t suits the organization, based on system load and user activity.
qad-analytics-core.browseBatchRefreshTimeLimit	Minimum number of minutes allowed between refreshes of the same KPI 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.

FRW KPI Scheduled Refresh Properties

Property	Purpose/Description	Default Value	Tuning Considerations
qad-analytics-financials.cache.kpis-frw.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-financials.cache.kpis-frw.scheduledRefresh.batchSize	Maximum number of refresh requests that are batched for processing in a single 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-financials.cache.kpis-frw.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-financials.cache.kpis-frw.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.	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 on-line user activity.
qad-analytics-financials.cache.kpis-frw.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-financials.cache.kpis-frw.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 2 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.

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.

Special Calendar Support

Fiscal Year and Quarter

For organizations that use a fiscal calendar different from the standard calendar, Action Centers 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 Enterprise 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 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 Files

The Action Center dashboard and visual definitions are stored in XML files inside the Logi Info web application, not in a database. 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 webapp. 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 sysadmin 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, it is suggested 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 Keyspace

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 backup in case of data loss. Instead, the keyspace should be rebuilt from its sources as described in the section 'Rebuilding the Cassandra Keyspace.'

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 backup in case of data loss. Instead, the cache will be rebuilt at the same time as the Cassandra keyspace (see above).

Log Files

Because much 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
```

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.

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* keyspace, which contains 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.

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 port number.

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

Because all the Query Service data is stored in the browses keyspace, make this keyspace the default for all subsequent CQL commands.

```
cqlsh> use browses;
```

Once the shell is running, it can 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 which browses are stored in the data lake, and for which domains or financials entities.

```
cqlsh:browses> describe tables;

mfg_wo251_10usa                       mfg_po042_20fra
mfg_ea023                             kpi_sales_analysis_2_21nlco
mfg_ic763_10usa                       mfg_po042_11can
mfg_po041_10usa                       kpi_sales_analysis_2_10usaco
mfg_po042_31aus                       mfg_ic752_10usa
mfg_po042_21nl                        mfg_gp823
mfg_wo254_10usa                       mfg_wo253_10usa
browsecopystatus                      kpi_sales_analysis_2_40brzco
mfg_ic763_11can                       mfg_wo257_10usa
kpi_sales_analysis_2_22ukco           mfg_wo252_10usa
kpi_sales_analysis_2_31ausco          kpi_sales_analysis_2_20fraco
mfg_po042_10usa                       mfg_ic763_12mex
kpi_sales_analysis_2_21itco           kpi_sales_analysis_2_31apcons
mfg_po042_30chn                       kpi_sales_analysis_2_12mexco
mfg_po042_12mex                       mfg_an001_10usaco
fin_bdinvoice_selectdinvoice_10usaco  kpi_sales_analysis_2_30chnco
mfg_wo258_10usa                       kpi_sales_analysis_2_11canco
kpi_sales_analysis_2_23gerco

cqlsh:browses>
```

Display status information about each browse

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

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

 browsecopy                                        | amountcopied | canceled |
failed                          | finished                        |
started                         | submitted
-------------------------------------------------+--------------+----------
+---------------------------------+---------------------------------
+---------------------------------+---------------------------------
```

```
                      sales analysis 2__11canco |          420 |     null
|                          null | 2018-10-02 15:33:53.135000+0000 | 2018-10-02 15:
33:39.119000+0000 | 2018-10-02 15:33:39.032000+0000
                    urn:browse:mfg:po042_11can_ |         3079 |     null
|                          null | 2018-09-27 18:30:57.756000+0000 | 2018-09-27 18:
30:02.782000+0000 | 2018-09-27 18:30:02.749000+0000
                      sales analysis 2__30chnco |          420 |     null
|                          null | 2018-10-02 15:34:31.645000+0000 | 2018-10-02 15:
33:55.682000+0000 | 2018-10-02 15:33:40.786000+0000
                    urn:browse:mfg:wo253_10usa_ |           55 |     null
|                          null | 2018-09-30 22:52:56.396000+0000 | 2018-09-30 22:
52:50.146000+0000 | 2018-09-30 22:52:50.120000+0000
                       urn:browse:mfg:ea023__ |          204 |     null
|                          null | 2018-10-03 15:56:04.636000+0000 | 2018-10-03 15:
55:59.573000+0000 | 2018-10-03 15:55:59.570000+0000
                    urn:browse:mfg:po042_12mex_ |            0 |     null | 2018-
09-27 18:30:27.326000+0000 |                              null | 2018-09-27 18:30:
05.105000+0000 | 2018-09-27 18:30:05.097000+0000
                    urn:browse:mfg:ic752_10usa_ |        14481 |     null
|                          null | 2018-10-03 15:58:13.972000+0000 | 2018-10-03 15:
57:47.554000+0000 | 2018-10-03 15:57:47.554000+0000
                    urn:browse:mfg:ic763_11can_ |        44128 |     null
|                          null | 2018-10-02 22:13:13.082000+0000 | 2018-10-02 22:
03:30.261000+0000 | 2018-10-02 22:03:28.292000+0000
                      sales analysis 2__20fraco |          389 |     null
|                          null | 2018-10-02 15:34:22.959000+0000 | 2018-10-02 15:
33:55.718000+0000 | 2018-10-02 15:33:40.672000+0000
                       sales analysis 2__21itco |            0 |     null
|                          null | 2018-10-02 15:34:46.948000+0000 | 2018-10-02 15:
34:34.274000+0000 | 2018-10-02 15:33:41.412000+0000
                      sales analysis 2__12mexco |          420 |     null
|                          null | 2018-10-02 15:35:01.220000+0000 | 2018-10-02 15:
34:34.276000+0000 | 2018-10-02 15:33:41.635000+0000
                      sales analysis 2__10usaco |          423 |     null
|                          null | 2018-10-02 15:35:13.076000+0000 | 2018-10-02 15:
34:49.723000+0000 | 2018-10-02 15:33:42.057000+0000
                    urn:browse:mfg:ic763_10usa_ |        43928 |     null
|                          null | 2018-10-02 22:12:49.377000+0000 | 2018-10-02 22:
03:30.393000+0000 | 2018-10-02 22:03:28.295000+0000
                    urn:browse:mfg:po042_31aus_ |         3460 |     null
|                          null | 2018-09-27 18:31:03.064000+0000 | 2018-09-27 18:
30:02.728000+0000 | 2018-09-27 18:30:02.715000+0000
                     urn:browse:mfg:po042_21nl_ |         3084 |     null
|                          null | 2018-09-27 18:30:58.849000+0000 | 2018-09-27 18:
30:02.903000+0000 | 2018-09-27 18:30:02.886000+0000
 urn:browse:fin:bdinvoice.selectdinvoice__10usaco |         3568 |     null
|                          null | 2018-09-20 22:40:54.409000+0000 | 2018-09-20 22:
40:40.132000+0000 | 2018-09-20 22:40:39.376000+0000
                    urn:browse:mfg:po042_30chn_ |         3442 |     null
|                          null | 2018-09-27 18:31:09.528000+0000 | 2018-09-27 18:
30:27.364000+0000 | 2018-09-27 18:30:05.685000+0000
                       sales analysis 2__31apcons |            0 |     null
|                          null | 2018-10-02 15:33:52.940000+0000 | 2018-10-02 15:
33:40.157000+0000 | 2018-10-02 15:33:39.989000+0000
                    urn:browse:mfg:wo254_10usa_ |          770 |     null
|                          null | 2018-09-30 22:53:18.329000+0000 | 2018-09-30 22:
52:53.597000+0000 | 2018-09-30 22:52:49.909000+0000
                      sales analysis 2__23gerco |          413 |     null
|                          null | 2018-10-02 15:34:46.593000+0000 | 2018-10-02 15:
34:25.448000+0000 | 2018-10-02 15:33:41.217000+0000
                    urn:browse:mfg:wo252_10usa_ |          252 |     null
|                          null | 2018-09-30 22:51:12.546000+0000 | 2018-09-30 22:
50:58.799000+0000 | 2018-09-30 22:50:57.107000+0000
                      sales analysis 2__31ausco |          422 |     null
|                          null | 2018-10-02 15:35:11.540000+0000 | 2018-10-02 15:
34:49.146000+0000 | 2018-10-02 15:33:41.946000+0000
                   urn:browse:mfg:an001__10usaco |           81 |     null
|                          null | 2018-09-21 01:36:20.386000+0000 | 2018-09-21 01:
36:19.496000+0000 | 2018-09-21 01:36:19.496000+0000
                         urn:browse:mfg:gp823__ |           20 |     null
|                          null | 2018-09-19 18:47:20.059000+0000 | 2018-09-19 18:
47:18.577000+0000 | 2018-09-19 18:47:17.808000+0000
                    urn:browse:mfg:po042_20fra_ |         3084 |     null
|                          null | 2018-09-27 18:30:58.027000+0000 | 2018-09-27 18:
30:02.435000+0000 | 2018-09-27 18:30:02.425000+0000
                       sales analysis 2__22ukco |          429 |     null
|                          null | 2018-10-02 15:34:31.620000+0000 | 2018-10-02 15:
34:00.432000+0000 | 2018-10-02 15:33:40.818000+0000
                    urn:browse:mfg:wo258_10usa_ |         4578 |     null
```

```
|                           null | 2018-09-30 22:55:44.899000+0000 | 2018-09-30 22:
54:56.478000+0000 | 2018-09-30 22:54:56.469000+0000
                      urn:browse:mfg:po042_10usa_ |         3116 |     null
|                           null | 2018-09-20 00:57:23.600000+0000 | 2018-09-20 00:
57:09.430000+0000 | 2018-09-20 00:57:08.737000+0000
                         sales analysis 2__21nlco |          384 |     null
|                           null | 2018-10-02 15:33:57.956000+0000 | 2018-10-02 15:
33:39.687000+0000 | 2018-10-02 15:33:39.598000+0000
                      urn:browse:mfg:wo257_10usa_ |          327 |     null
|                           null | 2018-09-30 22:55:07.534000+0000 | 2018-09-30 22:
54:56.205000+0000 | 2018-09-30 22:54:56.195000+0000
                      urn:browse:mfg:wo251_10usa_ |           18 |     null
|                           null | 2018-09-30 22:51:06.859000+0000 | 2018-09-30 22:
50:58.802000+0000 | 2018-09-30 22:50:57.104000+0000
                        sales analysis 2__40brzco |          386 |     null
|                           null | 2018-10-02 15:35:19.103000+0000 | 2018-10-02 15:
35:03.685000+0000 | 2018-10-02 15:33:42.380000+0000
                      urn:browse:mfg:po041_10usa_ |         2560 |     null
|                           null | 2018-09-20 00:55:45.600000+0000 | 2018-09-20 00:
55:36.902000+0000 | 2018-09-20 00:55:36.900000+0000
                      urn:browse:mfg:ic763_12mex_ |        43655 |     null
|                           null | 2018-10-02 22:12:59.577000+0000 | 2018-10-02 22:
03:34.330000+0000 | 2018-10-02 22:03:28.298000+0000

cqlsh:browses>
```

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

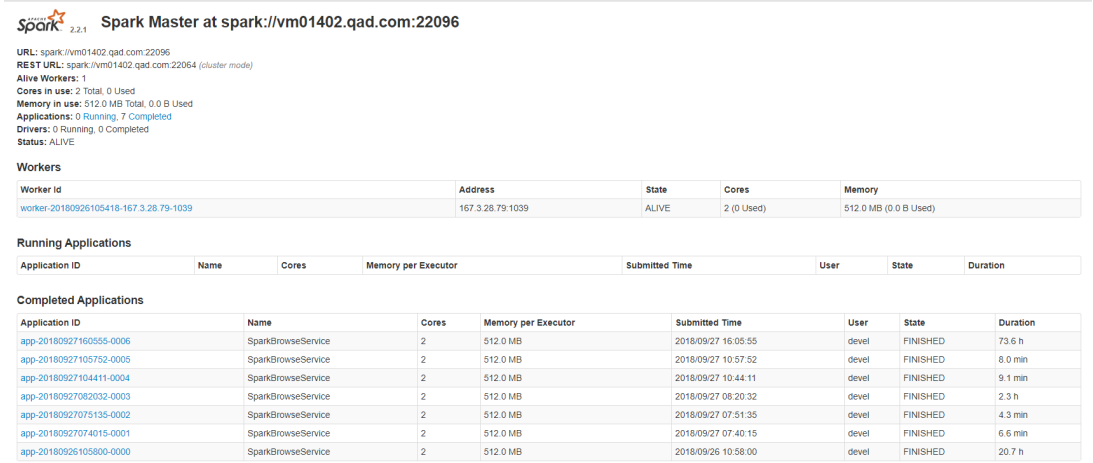
For more information, see the documentation at the Apache Spark web site.

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 WebUI 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 (0.0 B Used)

Running Applications

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

Completed Applications

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20180927160555-0006	SparkBrowseService	2	512.0 MB	2018/09/27 16:05:55	devel	FINISHED	73.6 h
app-20180927105752-0005	SparkBrowseService	2	512.0 MB	2018/09/27 10:57:52	devel	FINISHED	8.0 min
app-20180927104411-0004	SparkBrowseService	2	512.0 MB	2018/09/27 10:44:11	devel	FINISHED	9.1 min
app-20180927082032-0003	SparkBrowseService	2	512.0 MB	2018/09/27 08:20:32	devel	FINISHED	2.3 h
app-20180927075135-0002	SparkBrowseService	2	512.0 MB	2018/09/27 07:51:35	devel	FINISHED	4.3 min
app-20180927074015-0001	SparkBrowseService	2	512.0 MB	2018/09/27 07:40:15	devel	FINISHED	6.6 min
app-20180926105800-0000	SparkBrowseService	2	512.0 MB	2018/09/26 10:58:00	devel	FINISHED	20.7 h

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

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 webapp, 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.service, com.qad.analytics.core.util
Action Centers - Financials	com.qad.analytics.financials, com.qad.analytics.financials.mvc.controller.data, com.qad.analytics.financials.service, com.qad.analytics.financials.util
Query Service - Spark processing	com.qad.qracore.service.impl.SparkBrowseServiceImpl

Query Service - Cassandra processing	com.qad.qracore.service.impl.BrowseCassandraDataServiceImpl, com.qad.qracore.service.impl.BrowseDefinitionServiceImpl, com.qad.qracore.service.impl.CassandraCopyServiceImpl, 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 Info

When investigating problems in the display of visuals or grids inside the Action Centers, 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.

Rebuilding the Cassandra Keyspace

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 re-created and re-populated by processing the necessary browses against the current OpenEdge source databases.

To empty the contents of the Cassandra data lake and rebuild it from scratch, run the following YAB commands.

```
yab cassandra-stop
yab cassandra-rebuild
yab cassandra-start
```

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.

Checking the Financial Daemons

The Financial Report Writer (FRW) feature that supports Action Centers relies on background processes called daemons to continuously update the FRW KPI data as General Ledger entries are created. If some of these daemons are not running, the FRW KPIs and Action Centers may not reflect all current financial activity. In particular, the History Daemon and Cube Daemon are required.

The daemons are not automatically started when the environment is started using YAB. To check if the daemons are running, run the following YAB command.

```
yab daemon-status

                    daemon-status (13 tasks)                          [APPLY]
--------------------------------------------------------------------------------
1/13  daemon-xmldaemon-status                                STOPPED (1.591 s)
2/13  daemon-balancedaemon-status                            STOPPED (0.629 s)
3/13  daemon-budgetdaemon-status                             STOPPED (0.619 s)
4/13  daemon-crosscompanydaemon-status                       STOPPED (0.573 s)
5/13  daemon-eventdaemon-status                              STOPPED (0.627 s)
6/13  daemon-historydaemon-status                            STOPPED (0.621 s)
7/13  daemon-replicationdaemon-status                        STOPPED (0.621 s)
8/13  daemon-scandaemon-status                               STOPPED (0.621 s)
9/13  daemon-timeoutdaemon-status                            STOPPED (0.619 s)
10/13 daemon-cubedaemon-status                               STOPPED (0.619 s)
11/13 daemon-reportdaemon-status                             STOPPED (0.622 s)
12/13 daemon-batchdaemon-status                              STOPPED (0.618 s)
13/13 daemon-status                                          OK (1.933 s)
--------------------------------------------------------------------------------
```

To start the history and cube daemons, run the following YAB command.

```
yab daemon-historydaemon-start daemon-cubedaemon-start


   daemon-historydaemon-start daemon-cubedaemon-start (2 tasks)       [APPLY]
--------------------------------------------------------------------------------
1/2 daemon-historydaemon-start                               STARTED (18.434 s)
2/2 daemon-cubedaemon-start                                  STARTED (6.833 s)
--------------------------------------------------------------------------------
```

The daemons can also be monitored, started, and stopped from screens in the .NET UI. Daemon setup is described in the *QAD System Administration User Guide*.

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*.