Plan Qlik Sense deployments

Qlik Sense®
3.2
Copyright © 1993-2017 QlikTech International AB. All rights reserved.

Copyright © 1993-2017 QlikTech International AB. All rights reserved.

Qlik®, QlikTech®, Qlik Sense®, QlikView®, Sense® and the Qlik logo are trademarks which have been registered in multiple countries or otherwise used as trademarks by QlikTech International AB. Other trademarks referenced herein are the trademarks of their respective owners.

Contents

1 Introduction

In order to make the most of Qlik Sense, it is recommended to take the following into consideration when planning your deployment:

- Architecture: Qlik Sense features a distributed architecture that consists of one or more nodes (that is, server machines) that together form a site. One node assumes the role of central node, which is used as the central point of control.
- Deployment: Qlik Sense can be deployed in different ways to suit different needs. You can, for example, set up production sites and development sites.
- Backup and restore: It is recommended to back up your Qlik Sense site and the certificates used on the central node, so that they are available in case a restore is needed.
- Security: Sites can be deployed in a number of ways. Qlik Sense has therefore been designed to support security in many different ways.
- Log messages: The log messages produced by Qlik Sense provide important information that can be used to detect security incidents, operational problems, and policy violations.
- Licensing: The licensing in Qlik Sense is based on tokens, which are used to allocate access passes that allow users to access Qlik Sense. There are different types of access passes to choose from and each type corresponds to a specific consumption model for accessing Qlik Sense.

This document is derived from the online help for Qlik Sense. It is intended for those who want to read parts of the help offline or print pages easily, and does not include any additional information compared with the online help.

1.1 Conventions

The following conventions are used in the documentation for Qlik Sense.

Style coding

- Menu commands and dialog options are written in **bold**.
- File names and paths are written in *Italics*.
- Sample code is written in `Lucida Console`.

Environment variables

The paths used in the documentation for Qlik Sense may use environment variables. The variables and the equivalent paths in the Microsoft Windows operating system are listed below.

Environment variable	Microsoft Windows
%LocalAppData%	*C:\Users\<username>\AppData\Local*
%ProgramData%	*C:\ProgramData*
%ProgramFiles%	*C:\Program Files*
%UserProfile%	*C:\Users\<username>*

1.2 Additional documentation

Besides this document, the following related documentation is available for Qlik Sense:

- Install and upgrade Qlik Sense: Describes how to install Qlik Sense.
- Manage Qlik Sense sites: Describes how to manage a Qlik Sense site.

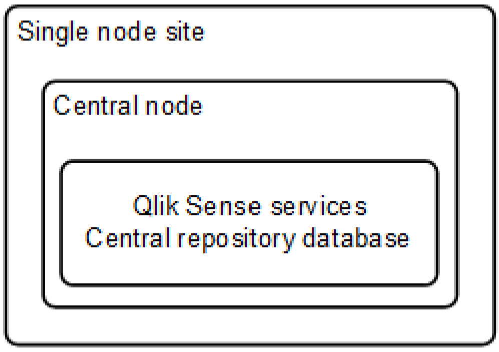
2 Architecture

The Qlik Sense architecture consists of a number of services that perform various activities in the software. The services can be distributed across one or more nodes , that is, server machines, that together form a site. One node assumes the role of central node, which is used as the central point of control and the other nodes can be configured to perform specific roles within the site such as adding capacity for users, or running reloads.

A site contains a common set of data in the form of Qlik Sense apps and configuration data, and contains at least one node.

2.1 Single node site

A single node site is the smallest site possible as it consists of a single node (that is, a single server machine), which is also the central node of the site.



See: *Deploying single node sites (page 39)*

2.2 Multi-node site

In a multi-node site, the site is spread out across two or more nodes that share the same set of data and license key. Multi-node sites can be used for many purposes:

- Add capacity
- Add resilience
- Move apps or workload onto a specific node
- Fit with customer network deployments

All content of the site is created and stored in a database and in files. You can chose one of two methods for storing this content within the site:

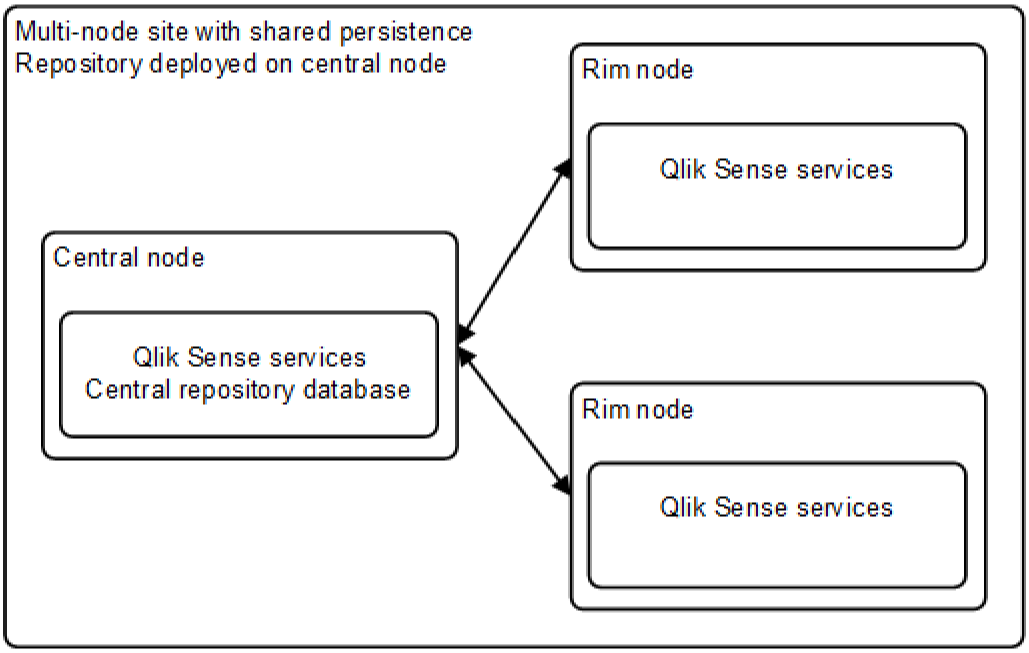
- **Shared persistence**

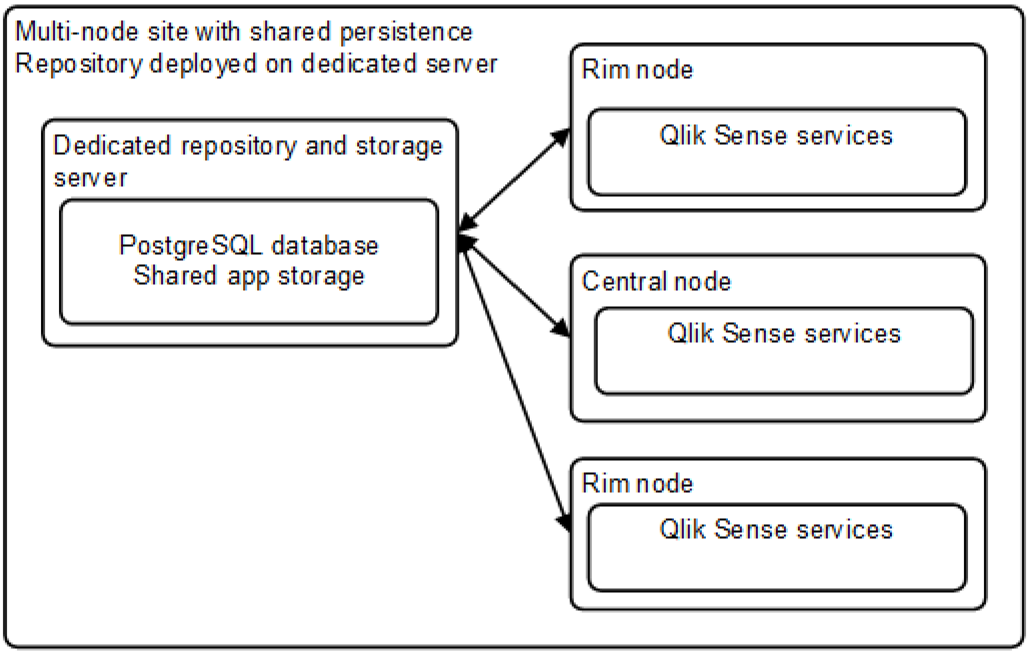
  In this mode a site has a single shared database and a single file store. We recommend this option for all new deployments of Qlik Sense.

- **Synchronized persistence**

  In this mode each server has its own local data base copy and set of files, and a sync mechanism keeps them in sync.
  It is possible to migrate an existing synchronized persistence deployment to shared persistence.

Shared persistence

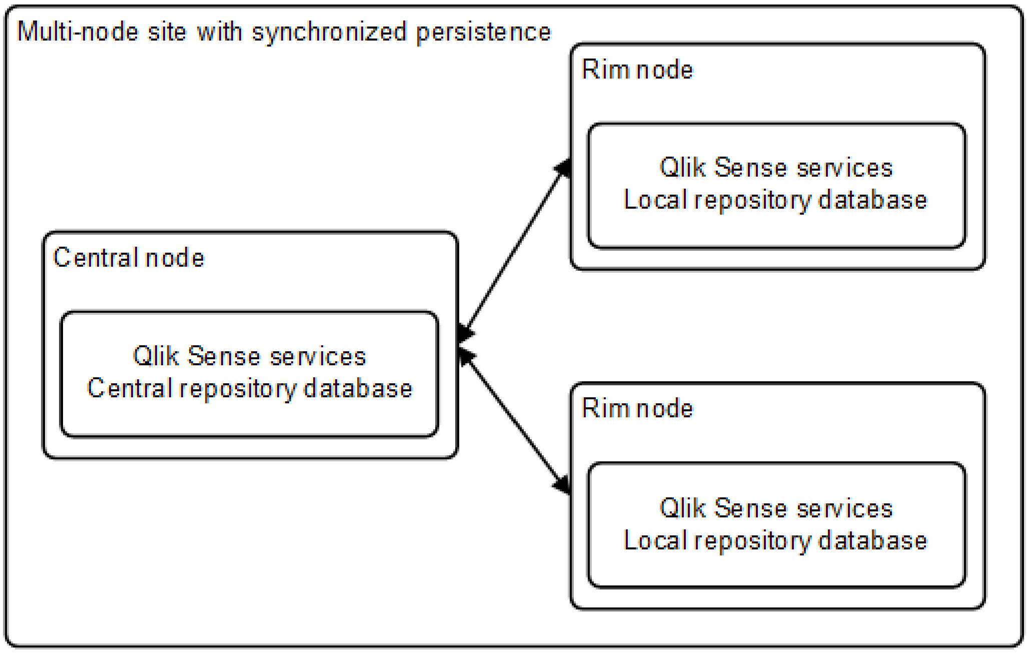

Multi-node site with shared persistence
Repository deployed on central node
Rim node
Qlik Sense services
Central node
Qlik Sense services
Central repository database
Rim node
Qlik Sense services



A common database and file share is made available to contain all of the data for all nodes. These can either be located on the central node, or can be placed onto dedicated infrastructure to provide higher performance and resilience.

All data and applications are available to all machines, and any changes that occur are immediately visible to all other nodes in the site. You can use load balancing rules to route users or reloads on particular apps to a specific node in the site.

Synchronized persistence


Multi-node site with synchronized persistence
Rim node
Qlik Sense services
Local repository database
Central node
Qlik Sense services
Central repository database
Rim node
Qlik Sense services
Local repository database

In a synchronized persistence multi-node site, each node has a local copy of the data that it needs to fulfill its role. Each node can read and write its local data and a synchronization mechanism in Qlik Sense distributes the changes to other nodes in the site.

One node is configured to be the central node, which is responsible for controlling the multi-node site. The central node is also the point through which the other nodes in the site synchronize their data.

The synchronization in a multi-node site with synchronized persistence is two-way:

- The central node requests updates from the rim nodes every 15 seconds.
- Each rim node initiates a synchronization session with the central node every 15 seconds.

When changes are made on each node, the resulting transactions are recorded in a transaction log. During the synchronization, the latest set of transactions from the log is sent to the other nodes and replayed.

The synchronization is not visible to the users, who can continue to work in their apps while new data is synchronized in the background.

See: *Deploying multi-node sites with synchronized persistence (page 45)*

2.3 Nodes

A Qlik Sense site consists of the following types of nodes:

- A central node: This is the minimum configuration; a site always includes a central node
- Zero or more rim nodes: Used to increase capacity and resilience

Each node in a site:

- Can perform a different role within the site
- Operates independently, which increases the system resilience, reduces maintenance, and increases the deployment flexibility
- Deploys a set of Qlik Sense services
  See: *Services (page 17)*

2.4 Clients

The clients are used to communicate and interact with Qlik Sense sites.

Hub

The hub is used to access and publish apps in Qlik Sense. The hub runs in a web browser, so no software installation is required.

Once established, the hub traffic only involves a rim node (unless the site is a single node site) and the hub.

Qlik Management Console

The Qlik Management Console (QMC) is used for configuration and administration of a Qlik Sense site.

The QMC only communicates logically with the central node. This means that:

- The QMC always uses the Qlik Sense Proxy Service (QPS) on the central node.
- For maximum performance within a multi-node site, it is recommended not to allow any user traffic on the central node.

2.5 Apps

A Qlik Sense app is a collection of reusable data items (measures, dimensions, and visualizations), sheets, and stories. It is a self-contained entity that includes the data to analyze in a structured data model.

The apps replace the documents that are used in QlikView.

Default storage

By default, apps are stored as follows:

- Repository database: Contains the app structure, including the paths to the binary files in the local file system. This data is referred to as entity data and is usually small in size.
- Local file system: Stores the app data as binary files. These files are referred to as binary data and the data model element of the files can be large in size. The files are by default stored in *%ProgramData%\Qlik\Sense\Apps*.

Portable format

An app can be stored in the local file system in the proprietary QVF format, which is a portable format.

A single app is stored as *<App name>.qvf*.

> *For an app to run in Qlik Sense, it must be stored in the repository database.*

2.6 Synchronization

The synchronization in a multi-node site with synchronized persistence is two-way:

- The central node requests updates from the rim nodes every 15 seconds.
- Each rim node initiates a synchronization session with the central node every 15 seconds.

When changes are made on each node, the resulting transactions are recorded in a transaction log. During the synchronization, the latest set of transactions from the log is sent to the other nodes and replayed.

The synchronization is not visible to the users, who can continue to work in their apps while new data is synchronized in the background.

Data to synchronize

There are two types of data that need to be synchronized:

- Entity data: The repository database contains the system configuration and all meta data about apps. This data is referred to as entity data and is usually small in size. The repository database is controlled by the Qlik Sense Repository Service (QRS).
- Binary data: The app data files contain the data models and app definitions. These files are referred to as binary data and the data model element of the files can be large in size. The app data files are controlled by the Qlik Sense Engine Service (QES).

Entity data synchronization

If the transaction log only contains entity data (that is, changes in the repository database), an entity data synchronization is performed. The changes are applied immediately in the repository database on the receiving node. If a conflict occurs, the latest transaction is used.

Example:

A user consumes a license on a rim node. The record is committed to the repository database and a transaction is recorded in the log. During the next synchronization, the central node asks the rim node for its latest transactions and applies them to its local database. The rest of the rim nodes get the same update from the central node during their next synchronization.

Binary data synchronization

If the transaction log contains binary data (that is, changes to app data files), a binary data synchronization, during which the receiving node obtains the updated data, is initiated. The entire app data file does not have to be copied, just the components that have changed. This means that small edits (for example, a new sheet in an app) are synchronized quickly and independently from a large edits (for example, a data model that is synchronized after a reload). During binary data synchronization, the nodes use peer-to-peer replication to speed up the synchronization of large apps and prevent network bottlenecks.

Example:

An app is reloaded on a rim node. During the next synchronization, the central node checks the transaction log and initiates a binary data synchronization. The rest of the rim nodes get the same update during their next synchronization with the central node. However, the rim nodes can obtain the binary data not just from the central node, but from any rim node that already has the updates.

2.7 Services

The Qlik Sense services, which run on the Microsoft Windows operating system, can be deployed in different ways on a node to suit different deployment purposes.

Services overview

The Qlik Sense services include:

- The Qlik Sense Repository Service (QRS) manages persistence and synchronization of apps and licensing, security, and service configuration data. The QRS is needed by all other Qlik Sense services to run and serve apps. It attaches to a repository database, and also manages the repository database synchronization in multi-node sites with synchronized persistence. In addition, the QRS stores the app structures and the paths to the binary files (that is, the app data stored in the local file system).
- In a default Qlik Sense installation, the Qlik Sense Repository Service (QRS) uses the Qlik Sense Repository Database (QRD) service to read and write data in the repository database. A PostgreSQL database is used by default.
- The Qlik Sense Proxy Service (QPS) manages site authentication, session handling, and load balancing.
- The Qlik Sense Scheduler Service (QSS) manages the scheduled reloads of apps as well as other types of reload triggering based on task events.
- The Qlik Sense Engine Service (QES) is the application service, which handles all application calculations and logic.

- The Qlik Sense Printing Service (QPR) manages export in Qlik Sense.
- The Qlik Sense Service Dispatcher (QSD) is a service controller that is used to launch and manage the following Qlik Sense services:
    - Broker Service: The Broker Service acts as an interface to and an intermediary between services started by the Qlik Sense Service Dispatcher (QSD).
    - Data Profiling Service: The Data Profiling Service is used to access and modify the app load model.
    - Hub Service: The Hub Service controls which content a user is allowed to see based on authorization.
    - Migration Service: The Migration Service ensures that your apps can be used in the currently installed version of Qlik Sense.
    - Capability Service: The Capability Service is used to handle Qlik Sense system feature configuration.
    - Converter Service: The Converter Service is used by the QlikView converter tool.

Deployment examples of Qlik Sense services on a node

- Complete: A node that includes all Qlik Sense services.
- Proxy: A node that manages Qlik Sense authentication, session handling, and load balancing. The QRS and QPS are installed.
- Engine: A node that provides the analytical power of Qlik Sense to the client. The QRS and QES are installed.
- Proxy and engine: A combined node that includes the QRS, QPS, and QES.
- Scheduler: A node that manages scheduled reloads of Qlik Sense apps and other types of reload triggering. The QRS, QSS, and QES are installed. In order to perform reloads the QSS requires the QES to be running on the same node.

Service dependencies

This section describes the dependencies related to the Qlik Sense services (for example, dependencies on the operating system and other software).

Repository database

The Qlik Sense Repository Service (QRS) connects to the repository database to store and retrieve data necessary for the Qlik Sense services on the node on which the QRS is running. In a default Qlik Sense installation, the Qlik Sense Repository Service (QRS) uses the Qlik Sense Repository Database (QRD) service to read and write data in the repository database. A PostgreSQL database is used by default.

File system

The file system stores the binary files for the Qlik Sense apps.

Directory service

The QRS and Qlik Sense Proxy Service (QPS) communicate with a configured directory service (for example, Microsoft Active Directory) using, for example, LDAP or ODBC.

Start and restart of services

Normally, the Qlik Sense services are started automatically on a node.

Start-up behavior

The Qlik Sense Repository Database (QRD) and Qlik Sense Repository Service (QRS) are started first.

When any other Qlik Sense service starts, it contacts its local QRS to get configuration parameters. If the service is not (yet) configured to run, it periodically checks back with the local QRS.

Manual start

If the services are started manually, make sure to start them in the following order:

a. Qlik Sense Repository Database (QRD)
b. Qlik Sense Service Dispatcher (QSD)
c. Qlik Sense Repository Service (QRS)
d. Qlik Sense Proxy Service (QPS), Qlik Sense Engine Service (QES), Qlik Sense Scheduler Service (QSS), and Qlik Sense Printing Service (QPR) in no specific order

The order is important because the QRS is dependent on the QRD and the rest of the services are dependent on the QRS.

Selecting the metrics to display

Proceed as follows to select which metrics to display for the Qlik Sense services in the Performance Monitor in Microsoft Windows:

1. Select **Start**>**Run**.
2. Enter *perfmon* and click **OK**.
3. Expand **Monitoring Tools** in the left panel.
4. Select **Performance Monitor**.

   The Performance Monitor is displayed in the right panel.
5. Click the **+** (plus) icon in the toolbar at the top of the Performance Monitor.

   The **Add Counters** dialog is displayed.
6. Select the machine to add counters from in the **Select counters from computer:** drop-down list.

   The **Available counters** list is populated with counters.
7. Locate the following counter sets in the **Available counters** list:
   - Qlik Sense Proxy Service
   - Qlik Sense Repository Service - REST API
   - Qlik Sense Repository Service - Synchronization
   - Qlik Sense Scheduler Service
8. Click the **+** (plus) sign next to a counter set to expand the set.
9. Select the counters to display in the Performance Monitor.

10. Click **Add >>** to add the counters in the Performance Monitor.

    The added counters are listed in the **Added counters** list.

11. Click **OK**.

    The added counters are displayed in the Performance Monitor.

Qlik Sense Repository Service

The Qlik Sense Repository Service (QRS) manages persistence and synchronization of apps and licensing, security, and service configuration data. The QRS is needed by all other Qlik Sense services to run and serve apps. It attaches to a repository database, and also manages the repository database synchronization in multi-node sites with synchronized persistence. In addition, the QRS stores the app structures and the paths to the binary files (that is, the app data stored in the local file system).

Paths

The following table lists the paths used by the Qlik Sense Repository Service (QRS).

Executable	*%ProgramFiles%\Qlik\Sense\Repository\Repository.exe*
Data	*%ProgramData%\Qlik\Sense\Repository*
Logs	*%ProgramData%\Qlik\Sense\Log\Repository* See: *Logging (page 121)*
Repository database	In a default Qlik Sense installation, the repository database is a specific instance of PostgreSQL that runs its own database cluster specifically for the repository. All files related to the repository database in a default Qlik Sense installation are stored in the following folder: *%ProgramData%\Qlik\Sense\Repository\PostgreSQL*

Bootstrap mode

You can use the following parameters to start the Qlik Sense Repository Service in bootstrap mode when you need to deploy Qlik Sense with a service account that does not have administrator privileges.

See: Running Qlik Sense as a user without local Administrator privileges

- `-bootstrap`
  Use this parameter to start Qlik Sense Repository Service in bootstrap mode.
- `-bootstrap=install`
  Use this parameter to start Qlik Sense Repository Service in bootstrap mode when installing.
- `-bootstrap=uninstall`
  Use this parameter when uninstalling Qlik Sense.
- `-iscentral`
  Use this flag in addition to the bootstrap flag when installing or configuring a central node.

Do the following:

1. Stop Qlik Sense services except Qlik Sense Repository Database.
2. Run `repository.exe -bootstrap` from an elevated command prompt.
3. Start Qlik Sense services.

Metrics

This section lists the metrics related to the Qlik Sense Repository Service (QRS).

See: *Selecting the metrics to display (page 19)*

REST API metrics

The following metrics are available in the Performance Monitor in Microsoft Windows:

- Number of DELETE calls
- Number of GET calls
- Number of POST calls
- Number of PUT calls
- Number of HTTP status 200 (OK)
- Number of HTTP status 201 (Created)
- Number of HTTP status 400 (Bad request)
- Number of HTTP status 401 (Unauthorized)
- Number of HTTP status 403 (Forbidden)
- Number of HTTP status 406 (Not acceptable)
- Number of HTTP status 409 (Conflict)
- Number of HTTP status 415 (Unsupported media type)
- Number of HTTP status 500 (Internal server error)
- Number of HTTP status 503 (Service unavailable)

Synchronization metrics

The following metrics are available in the Performance Monitor in Microsoft Windows:

- Number of synchronization sessions
- Number of synchronization clients
- Synchronization client queue

Qlik Sense Repository Database

In a default Qlik Sense installation, the Qlik Sense Repository Service (QRS) uses the Qlik Sense Repository Database (QRD) service to read and write data in the repository database. A PostgreSQL database is used by default.

Paths

The following table lists the paths used by the Qlik Sense Repository Database (QRD) service.

Executable	In a default Qlik Sense installation, the repository database is a specific instance of PostgreSQL that runs its own database cluster specifically for the repository. The QRD spawns the PostgreSQL executable that is located in the following folder: *%ProgramFiles%\Qlik\Sense\Repository\PostgreSQL\<database version>\bin*
Data	*%ProgramData%\Qlik\Sense\Repository\PostgreSQL*
Logs	There are no logs for the QRD service.

Qlik Sense Proxy Service

The Qlik Sense Proxy Service (QPS) manages site authentication, session handling, and load balancing.

Paths

The following table lists the paths used by the Qlik Sense Proxy Service (QPS).

Executable	*%ProgramFiles%\Qlik\Sense\Proxy\Proxy.exe*
Data	*%ProgramData%\Qlik\Sense\Proxy*
Logs	*%ProgramData%\Qlik\Sense\Log\Proxy* See: *Logging (page 121)*

Bootstrap mode

You can use the following parameters to start the Qlik Sense Proxy Service in bootstrap mode when you need to deploy Qlik Sense with a service account that does not have administrator privileges.

See: Running Qlik Sense as a user without local Administrator privileges

- `-bootstrap`
  Use this parameter to start Qlik Sense Proxy Service in bootstrap mode.
- `-bootstrap=install`
  Use this parameter to start Qlik Sense Proxy Service in bootstrap mode when installing.
- `-bootstrap=uninstall`
  Use this parameter when uninstalling Qlik Sense.

Do the following:

1. Stop Qlik Sense services.
2. Run `proxy.exe -bootstrap` from an elevated command prompt.
3. Start Qlik Sense services.

Metrics

This section lists the metrics related to the Qlik Sense Proxy Service (QPS). The following metrics are available in the Performance Monitor in Microsoft Windows:

See: *Performance log (page 144)*

See: *Selecting the metrics to display (page 19)*

- ActiveConnections: The number of active connections (in any form or shape) from the client.

  A connection is a stream (that is, a socket) between a Qlik Sense client and the Qlik Sense Proxy Service (QPS). This stream is often connected to another stream, which runs from the QPS to the Qlik Sense Repository Service (QRS) or the Qlik Sense Engine Service (QES). The two streams allow the client to communicate with the QRS or the QES.
- ActiveStreams: The number of active data streams (that is, sockets), either from the browser to the QPS or from the QPS to the QRS or the QES.
- ActiveSessions: The number of active sessions in the QPS.

  A Qlik Sense user gets a proxy session when the user has been authenticated. The session is terminated after a certain period of inactivity.
- LoadBalancingDecisions: The number of users who currently have at least one engine session.
- PrintingLoadBalancingDecisions: The number of users who have been load balanced to the Qlik Sense Printing Service (QPR).
- Tickets: The number of issued login tickets that have not yet been consumed.
- ActiveClientWebsockets: The number of active WebSockets between the client and the QPS.
- ActiveEngineWebsockets: The number of active WebSockets between the QPS and the target Qlik Sense service.

The metrics are also available as entries in the Performance log for the QPS.

Qlik Sense Scheduler Service

The Qlik Sense Scheduler Service (QSS) manages the scheduled reloads of apps as well as other types of reload triggering based on task events.

Tasks

Tasks are used to perform a wide variety of operations and can be chained together in any arbitrary pattern. The tasks are handled by the Qlik Sense Scheduler Service (QSS) and managed in the Qlik Management Console (QMC).

Reload

The reload task is used to fully reload the data in an app from the source. Any old data is discarded.

Sync

Within a multi-node site with synchronized persistence, one instance of the Qlik Sense Repository Service (QRS) runs on each node. The QRS running on the central node is considered to be the master. The master QRS has direct access to the central repository database, whereas the other QRSs only have access to a local repository database on the node where they are running. The master QRS synchronizes the central repository database and the local repository databases.

The sync task is used to schedule the synchronization of the central repository database and the local repository databases within a multi-node site.

Paths

The following table lists the paths used by the Qlik Sense Scheduler Service (QSS).

Executable	*%ProgramFiles%\Qlik\Sense\Scheduler\Scheduler.exe*
Data	-
Logs	*%ProgramData%\Qlik\Sense\Log\Scheduler* See: *Logging (page 121)*

Bootstrap mode

You can use the following parameters to start the Qlik Sense Scheduler Service in bootstrap mode when you need to deploy Qlik Sense with a service account that does not have administrator privileges.

See: Running Qlik Sense as a user without local Administrator privileges

- `-bootstrap`
  Use this parameter to start Qlik Sense Scheduler Service in bootstrap mode.
- `-bootstrap=install`
  Use this parameter to start Qlik Sense Scheduler Service in bootstrap mode when installing.
- `-bootstrap=uninstall`
  Use this parameter when uninstalling Qlik Sense.

Do the following:

1. Stop Qlik Sense services.
2. Run `scheduler.exe -bootstrap` from an elevated command prompt.
3. Start Qlik Sense services.

Metrics

This section lists the metrics related to the Qlik Sense Scheduler Service (QSS). The following metrics are available in the Performance Monitor in Microsoft Windows:

See: *Selecting the metrics to display (page 19)*

- Number of connected slaves
- Number of Qlik Sense Engine Service (QES) instances that are running on a slave (this metric is only available on the node where the QES instances run)
- Number of running processes
- Number of running tasks as understood by the master
- Number of running tasks on the slave
- Number of task messages that have been dispatched by the slave
- Number of task messages that have been received by the master

- Number of task retries
- Number of tasks that have completed successfully when executed by the slave
- Number of tasks that have failed when executed by the slave
- Number of tasks that the master has acknowledged as completed
- Number of tasks that the master has acknowledged as failed
- Number of times that the settings have been updated
- Number of tasks that have attempted to start
- Number of tasks that have attempted to stop

Qlik Sense Engine Service

The Qlik Sense Engine Service (QES) is the application service, which handles all application calculations and logic.

Paths

The following table lists the paths used by the Qlik Sense Engine Service (QES).

Executable	*%ProgramFiles%\Qlik\Sense\Engine\Engine.exe*
Data	*%ProgramData%\Qlik\Sense\Engine*
Logs	*%ProgramData%\Qlik\Sense\Log\Engine* See: *Logging (page 121)*
Configuration	*%ProgramData%\Qlik\Sense\Engine\Settings.ini* This file contains the QES settings. The file is created when the service first runs.

Qlik Sense Printing Service

The Qlik Sense Printing Service (QPR) manages export in Qlik Sense.

Paths

The following table lists the paths used by the Qlik Sense Printing Service (QPR).

Executable	*%ProgramFiles%\Qlik\Sense\Printing\Printing.exe*
Data	*%ProgramData%\Qlik\Sense\Printing*
Logs	*%ProgramData%\Qlik\Sense\Log\Printing* See: *Logging (page 121)*

Qlik Sense Service Dispatcher

The Qlik Sense Service Dispatcher (QSD) is a service controller that is used to launch and manage the following Qlik Sense services:

- Broker Service: The Broker Service acts as an interface to and an intermediary between services started by the Qlik Sense Service Dispatcher (QSD).
- Data Profiling Service: The Data Profiling Service is used to access and modify the app load model.
- Hub Service: The Hub Service controls which content a user is allowed to see based on authorization.
- Migration Service: The Migration Service ensures that your apps can be used in the currently installed version of Qlik Sense.
- Capability Service: The Capability Service is used to handle Qlik Sense system feature configuration.
- Converter Service: The Converter Service is used by the QlikView converter tool.

Paths

The following table lists the paths used by the Qlik Sense Service Dispatcher (QSD) and the services that are launched and managed by the QSD.

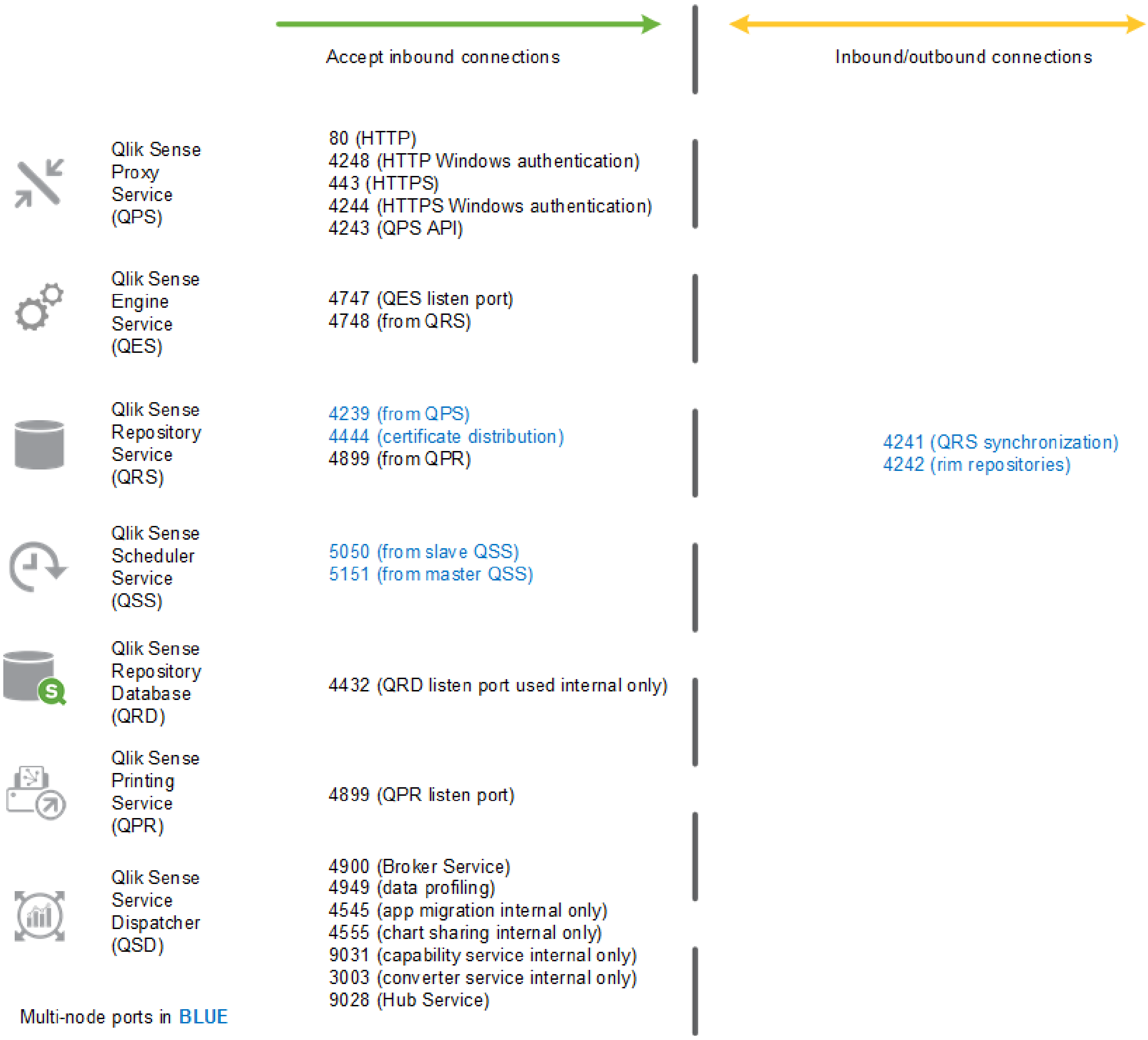
Executables	• QSD: *%ProgramFiles%\Qlik\Sense\ServiceDispatcher\ServiceDispatcher.exe* • Services that are launched and managed by the QSD: *%ProgramFiles%\Qlik\Sense\ServiceDispatcher\node\node.exe*
Logs	• Broker Service: *%ProgramData%\Qlik\Sense\Log\BrokerService* • Data Profiling Service: *%ProgramData%\Qlik\Sense\Log\DataProfiling* • Hub Service: *%ProgramData%\Qlik\Sense\Log\HubService* • Migration Service: *%ProgramData%\Qlik\Sense\Log\AppMigration* • Capability Service: *%ProgramData%\Qlik\Sense\Log*\CapabilityService See: *Logging (page 121)*

2.8 Ports

Qlik Sense uses ports for communication between web browsers (users) and proxies, and between services in both internal and multi-node deployments.

Ports overview synchronized persistence

This section provides an overview of the ports that are used in a Qlik Sense deployment with synchronized persistence.

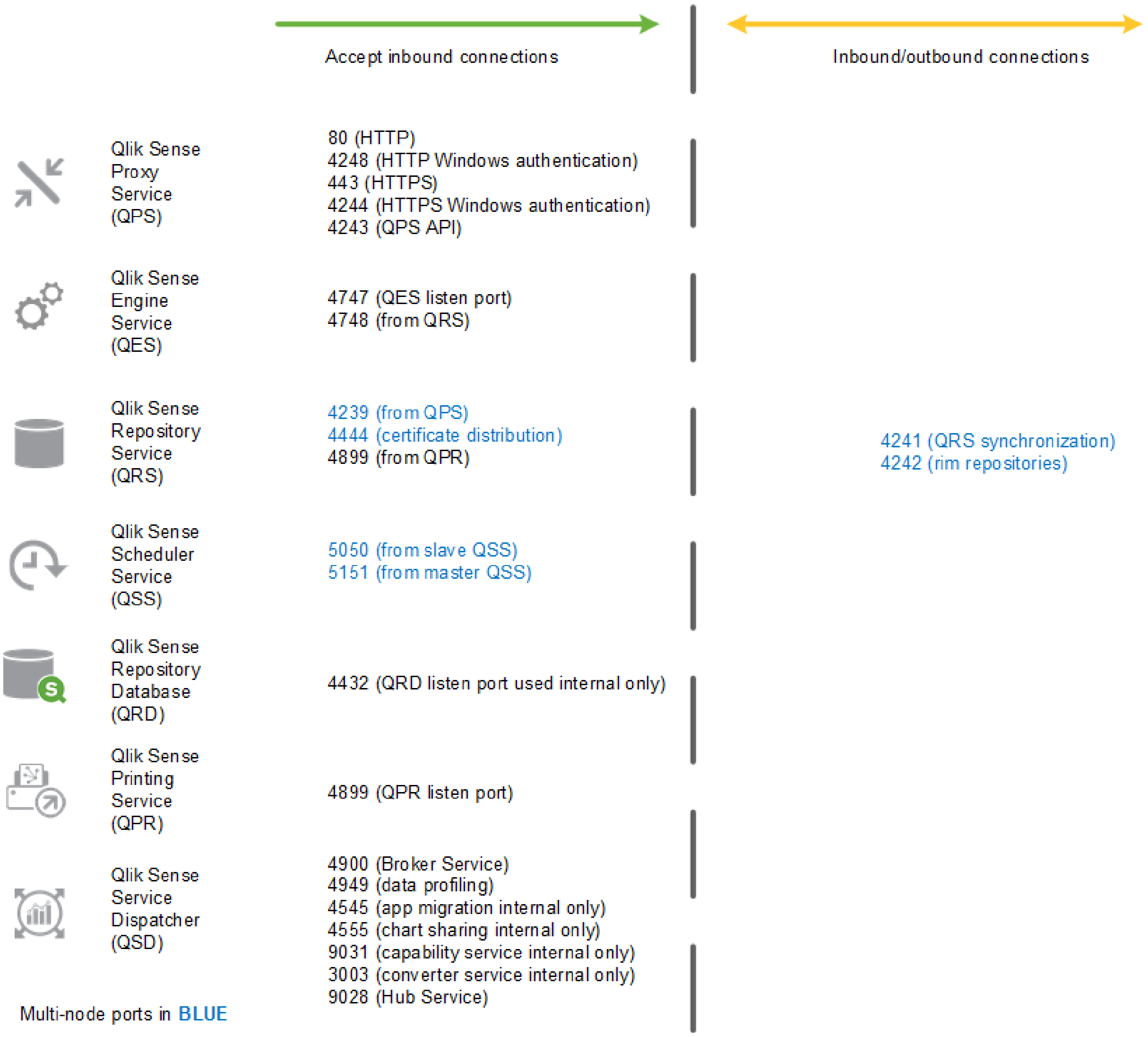


	Accept inbound connections	Inbound/outbound connections
Qlik Sense Proxy Service (QPS)	80 (HTTP) 4248 (HTTP Windows authentication) 443 (HTTPS) 4244 (HTTPS Windows authentication) 4243 (QPS API)	
Qlik Sense Engine Service (QES)	4747 (QES listen port) 4748 (from QRS)	
Qlik Sense Repository Service (QRS)	4239 (from QPS) 4444 (certificate distribution) 4899 (from QPR)	4241 (QRS synchronization) 4242 (rim repositories)
Qlik Sense Scheduler Service (QSS)	5050 (from slave QSS) 5151 (from master QSS)	
Qlik Sense Repository Database (QRD)	4432 (QRD listen port used internal only)	
Qlik Sense Printing Service (QPR)	4899 (QPR listen port)	
Qlik Sense Service Dispatcher (QSD)	4900 (Broker Service) 4949 (data profiling) 4545 (app migration internal only) 4555 (chart sharing internal only) 9031 (capability service internal only) 3003 (converter service internal only) 9028 (Hub Service)	

Multi-node ports in BLUE

Ports overview shared persistence

This section provides an overview of the ports that are used in a Qlik Sense deployment with shared persistence.



	Accept inbound connections	Inbound/outbound connections
Qlik Sense Proxy Service (QPS)	80 (HTTP) 4248 (HTTP Windows authentication) 443 (HTTPS) 4244 (HTTPS Windows authentication) 4243 (QPS API)	
Qlik Sense Engine Service (QES)	4747 (QES listen port) 4748 (from QRS)	
Qlik Sense Repository Service (QRS)	4239 (from QPS) 4444 (certificate distribution) 4899 (from QPR)	4241 (QRS synchronization) 4242 (rim repositories)
Qlik Sense Scheduler Service (QSS)	5050 (from slave QSS) 5151 (from master QSS)	
Qlik Sense Repository Database (QRD)	4432 (QRD listen port used internal only)	
Qlik Sense Printing Service (QPR)	4899 (QPR listen port)	
Qlik Sense Service Dispatcher (QSD)	4900 (Broker Service) 4949 (data profiling) 4545 (app migration internal only) 4555 (chart sharing internal only) 9031 (capability service internal only) 3003 (converter service internal only) 9028 (Hub Service)	

Multi-node ports in **BLUE**

Ports used internally within a node

The ports in the following table are used between Qlik Sense services that run on the same node. In most cases, the ports do not have to be open through any firewalls.

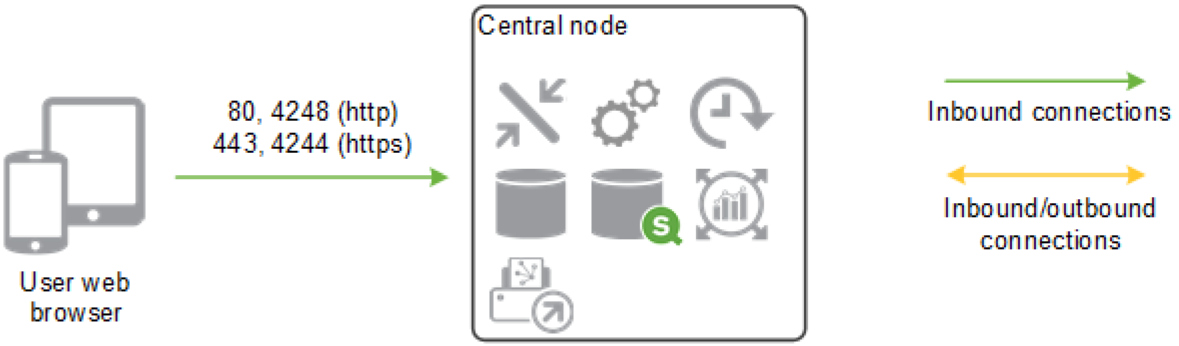
Service	Port	Direction	Purpose
Converter Service	3003	Internal	This port is used by the Converter Service which is utilized by QlikView converter.
QPS	4243	Inbound	Qlik Sense Proxy Service (QPS) REST API listen port. If web ticketing is used for security, this port is used by the software or service that requests tickets for users. If the software or service is remote, this port needs to be open to the location from which it is called.

QRD	4432	Internal/Bi-directional	Default listen port for the Qlik Sense Repository Database (QRD). It needs to be configured differently depending on which persistence model you use. **Synchronized persistence** The port is used to listen for connections from the Qlik Sense Repository Service (QRS) and is only for internal use. **Shared persistence** The port is used to listen for connections from the Qlik Sense Repository Service (QRS) and needs to be bi-directional on the node where the database is installed.
Migration Service	4545	Internal	This port is used by the Migration Service for app migration purposes. The service is launched and managed by the Qlik Sense Service Dispatcher (QSD) when required. The Migration Service only runs on the central node.
Chart Sharing Service	4555	Internal	This port is used by the Chart Sharing Service for chart sharing between Qlik Sense users. The service is launched and managed by the Qlik Sense Service Dispatcher (QSD) when required. This port uses HTTPS for communication.
QRS	4570	Internal	Certificate password verification port, only used within multi-node sites by Qlik Sense Repository Services (QRSs) on rim nodes to receive the password that unlocks a distributed certificate. The port can only be accessed from localhost and it is closed immediately after the certificate has been unlocked. The communication is always unencrypted.
QES	4748	Internal	This callback port is used by the Qlik Sense Repository Service (QRS) for sending HTTP events to the Qlik Sense Engine Service (QES).
Broker Service	4900	Internal	Default listen port for the Broker Service.
Hub Service	9028	Internal	Default listen port for the Hub Service.
Capability Service	9031	Internal	This port is used by the Capability Service to handle Qlik Sense system feature configuration.

Ports used between user web browsers and proxies

The default ports are exposed to the Qlik Sense users and need to be open through any firewalls to each Qlik Sense Proxy Service (QPS) in the site.

Service	Port	Direction	Purpose
QPS	443	Inbound	Inbound user web traffic when using HTTPS.
QPS	4244	Inbound	Authentication port when using Windows authentication over HTTPS.
QPS	80	Inbound	Inbound user web traffic when using HTTP (optional).
QPS	4248	Inbound	Authentication port when using Windows authentication over HTTP (optional).



Ports used between nodes and Qlik Sense services

The ports in this section are used for communication between the Qlik Sense services.

In a single node site, all ports listed in this section are used by the various services, but do not need access through firewalls.

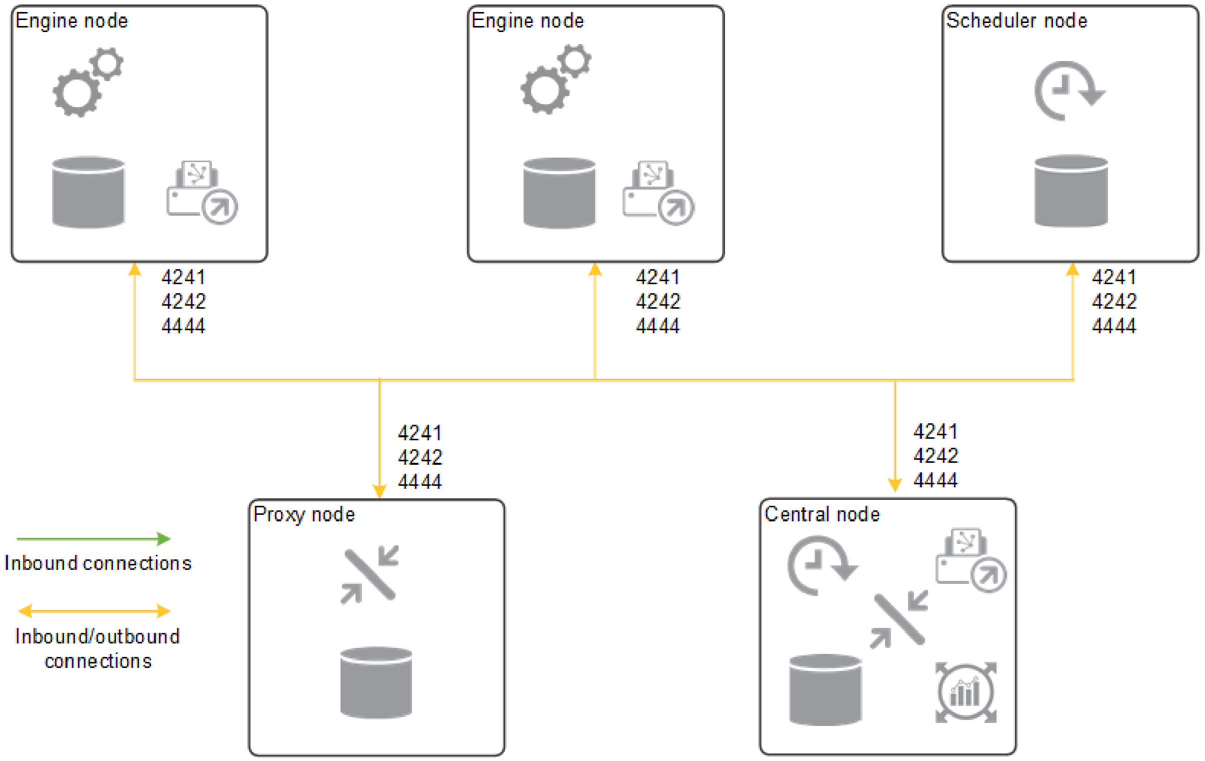
In a multi-node site, the ports in use vary depending on the services installed and running on each node. The ports need to be open in any firewalls between the nodes, but do not have to be open to the Qlik Sense users.

Minimum ports used for communication in multi-node sites

The following ports must always be open between the nodes in a multi-node site. The ports must be open to allow for synchronization, service health, and some specific operations.

Service	Port	Direction	Purpose
QRS	4241	Bi-directional between all nodes	Communication port within multi-node sites for Qlik Sense Repository Service (QRS)-to-QRS synchronization. This port uses HTTPS for communication. This port is not used in a shared persistence deployment.
QRS	4242	Bi-directional between the central node and all proxy nodes	This port is used for a number of operations including new user registration.

QRS	4444	Between the central node and all rim nodes	This port has two functions: • Security distribution port, only used by Qlik Sense Repository Services (QRSs) on rim nodes to receive a certificate from the master QRS on the central node. The communication is always unencrypted, but the transferred certificate package is password-protected. • Qlik Sense Repository Service (QRS) state port, used to fetch the state of a QRS in a Qlik Sense site. The state is fetched using *http://localhost:4444/status/servicestate*. The returned state is one of the following: ◦ 0: Initializing. Once the node has been initialized, the node state changes into one of the other states. ◦ 1: Certificates not installed. There are no certificates installed on the node. The node stays in this state until it has received the certificate and the certificate password. ◦ 2: Running. The node is up and running and all APIs have been initiated.



Ports used between master and slave schedulers

The ports in the following table are used when a slave Qlik Sense Scheduler Service (QSS) is used.

Service	Port	Direction	Purpose
QSS	5050	Inbound (from scheduler nodes only)	This port is used by the master QSS on the central node to issue commands to and receive replies from slave QSS nodes.
QSS	5151	Inbound (from the central node only)	A slave QSS runs on a slave scheduler node and is accessed only by the master QSS on the central node.

Ports used between a proxy node and an engine node

The ports in the following table define the minimum needed to allow regular user traffic and load balancing between a proxy node and an engine node.

Service	Port	Direction	Purpose
QES	4747	Inbound (from proxy nodes)	Qlik Sense Engine Service (QES) listen port. This is the main port used by the QES. The port is used via the Qlik Sense Proxy Service (QPS) for communication with the Qlik Sense clients.
QRS	4239	Inbound (from proxy nodes)	Qlik Sense Repository Service (QRS) WebSocket port. The port is used via the Qlik Sense Proxy Service (QPS) by the Qlik Sense hub to obtain apps and stream lists.
QRS	4242	Inbound (from proxy nodes)	Qlik Sense Repository Service (QRS) REST API listen port. This port is mainly accessed by local Qlik Sense services. However, the port must be open to all proxy nodes in a multi-node site to deliver images and static content.
Data Profiling Service	4949	Inbound (from proxy nodes)	This port is used by the Data Profiling Service when accessing and modifying the application load model. The service is launched and managed by the Qlik Sense Service Dispatcher (QSD) when required. The port is access via the Qlik Sense Proxy Service (QPS).
Broker Service	4900	Inbound (from proxy nodes)	Default listen port for the Broker Service.
Hub Service	9028	Inbound (from proxy nodes)	Default listen port for the Hub Service.

Ports used between a proxy node and the printing service

The Qlik Sense Printing Service (QPR) may be installed on the same node as other services or on a separate node. The ports in the following table must be accessible between a QPS and all QPRs to which the QPS can load balance traffic.

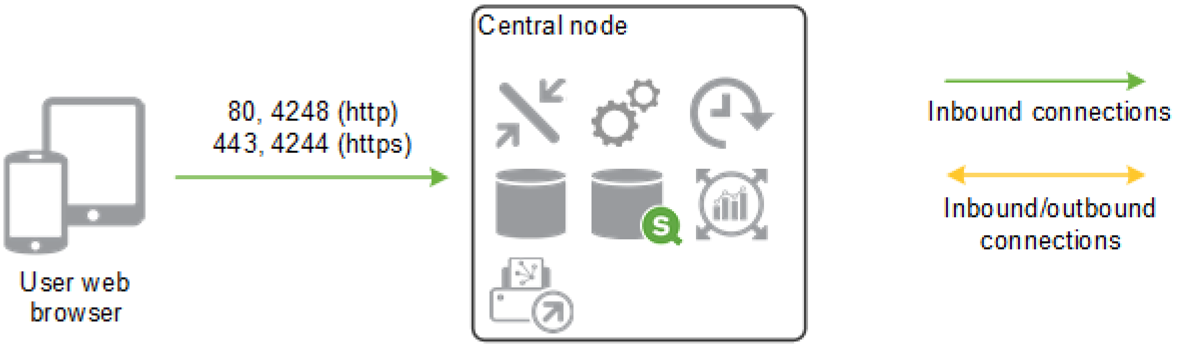
Service	Port	Direction	Purpose
QPR	4899	Inbound (from proxy nodes)	Qlik Sense Printing Service (QPR) port. This port is used for printed export in Qlik Sense. The port is accessed by any node that runs a QPS.

Ports examples

This section provides examples of the ports that are used in different Qlik Sense deployments with synchronized persistence.

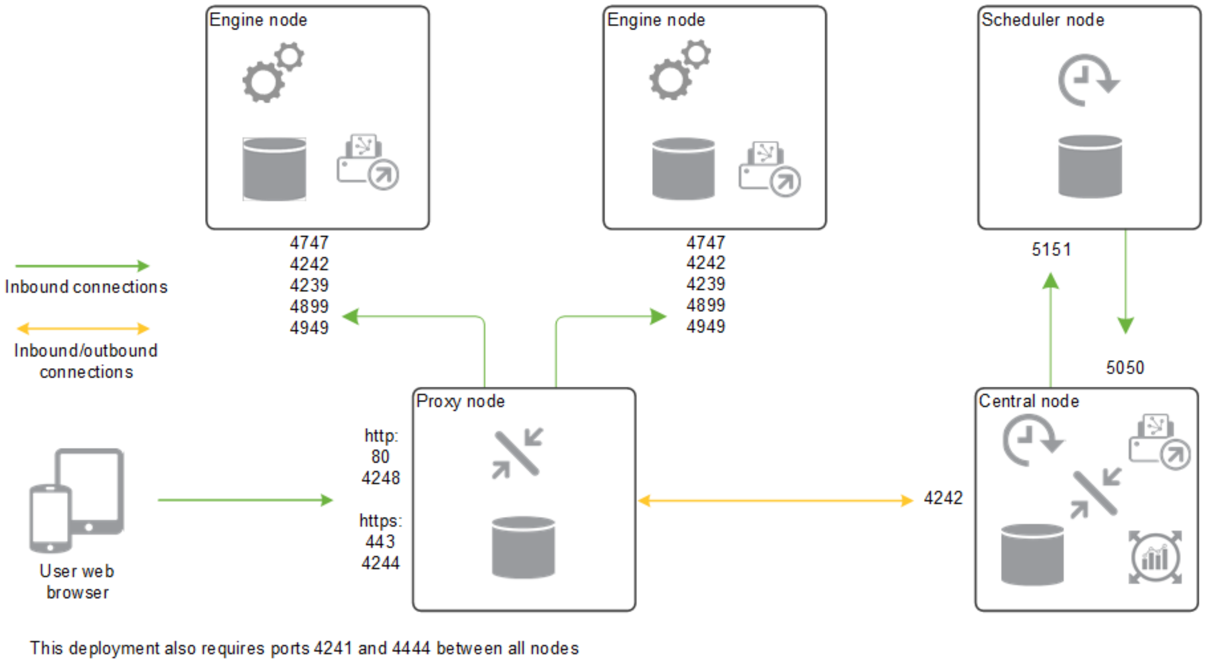
Single node site

This example shows the ports that are used in a single node site.



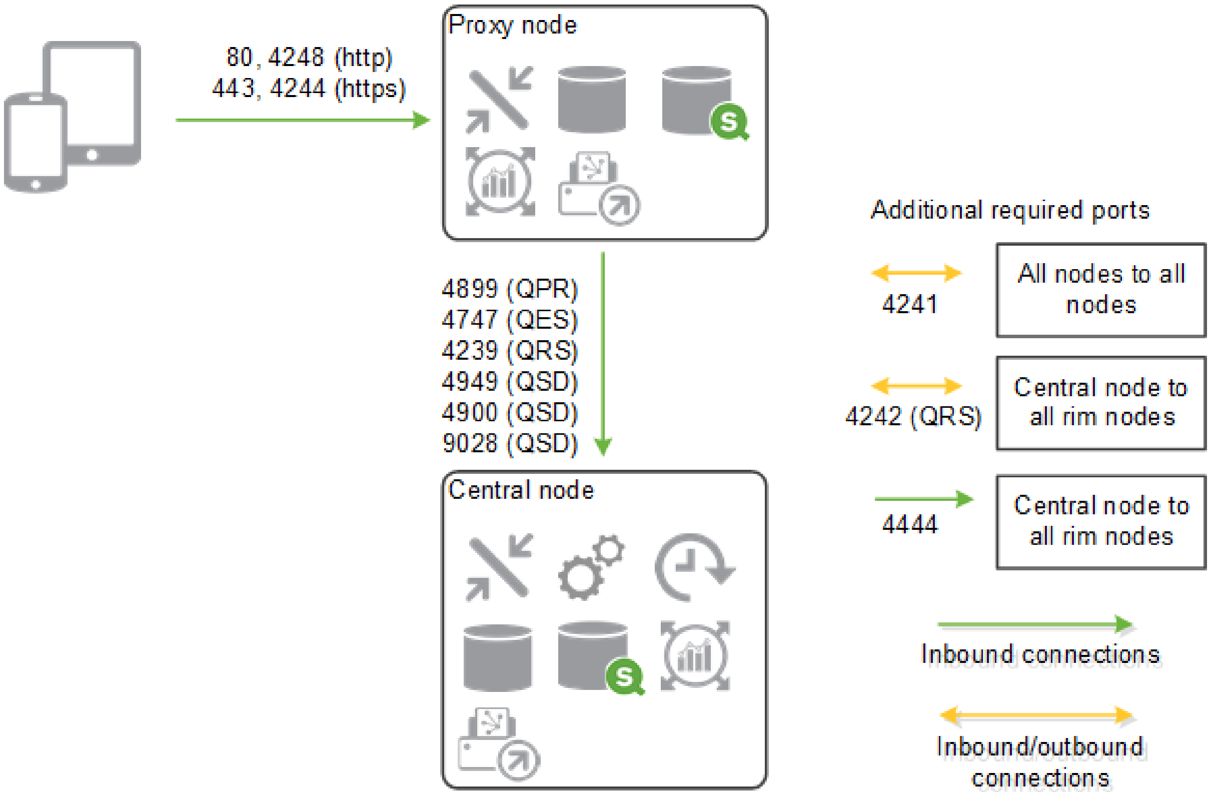
Multi-node site

The following is an example of the ports that are used in a multi-node site that consists of five nodes.



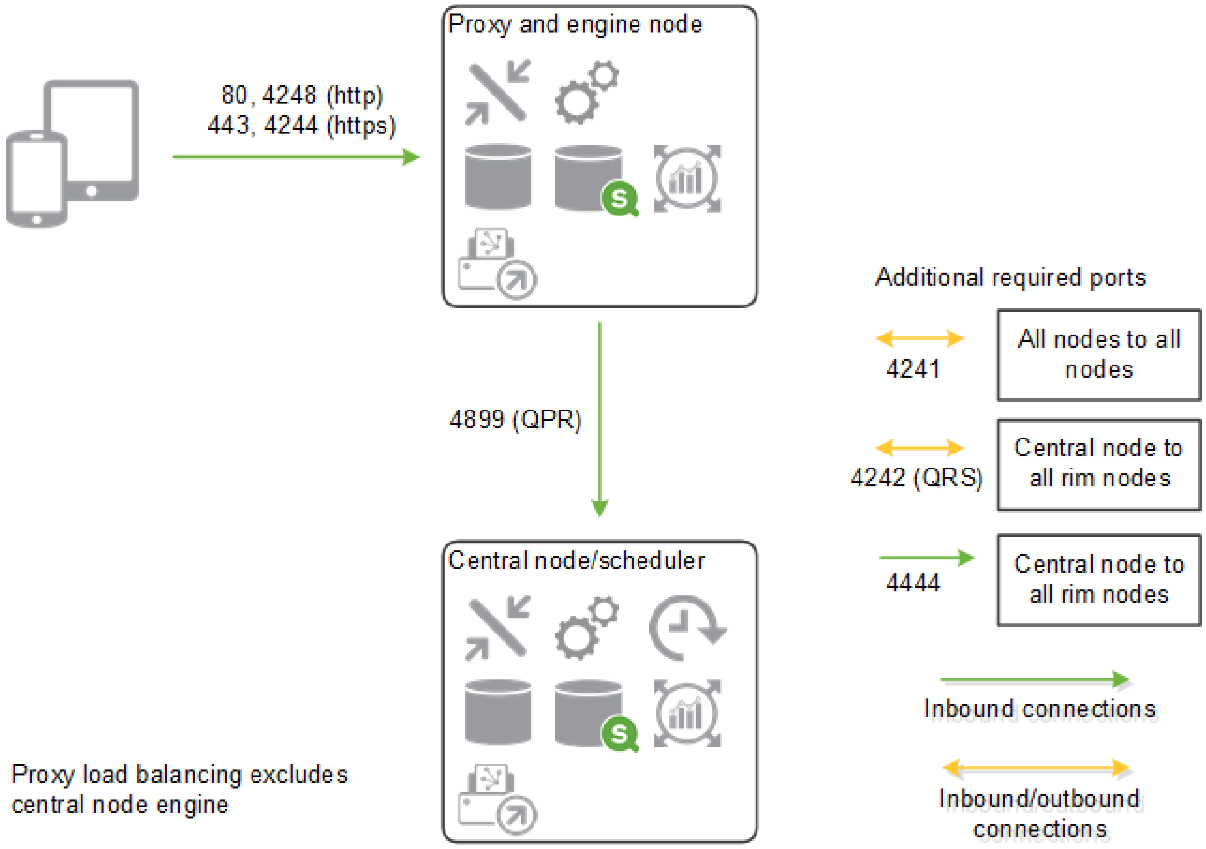
Proxy node in demilitarized zone

This example shows the ports that are used in a multi-node site when deploying a proxy node in a demilitarized zone.



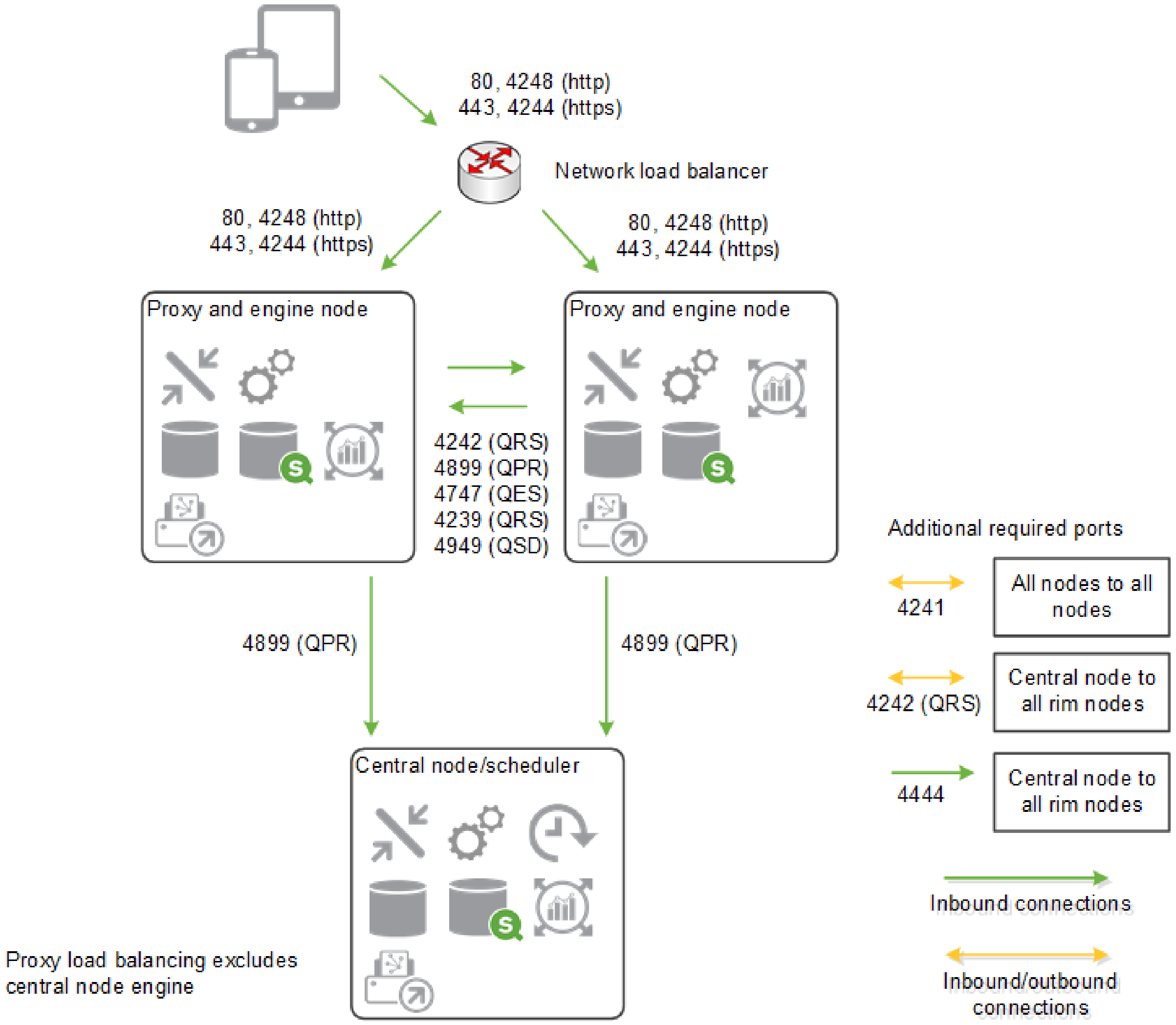
Separate proxy and engine node

This example shows the ports that are used in a multi-node site when deploying a separate proxy and engine node.


Proxy and engine node
80, 4248 (http)
443, 4244 (https)
4899 (QPR)
Central node/scheduler
Proxy load balancing excludes central node engine
Additional required ports
4241
All nodes to all nodes
4242 (QRS)
Central node to all rim nodes
4444
Central node to all rim nodes
Inbound connections
Inbound/outbound connections

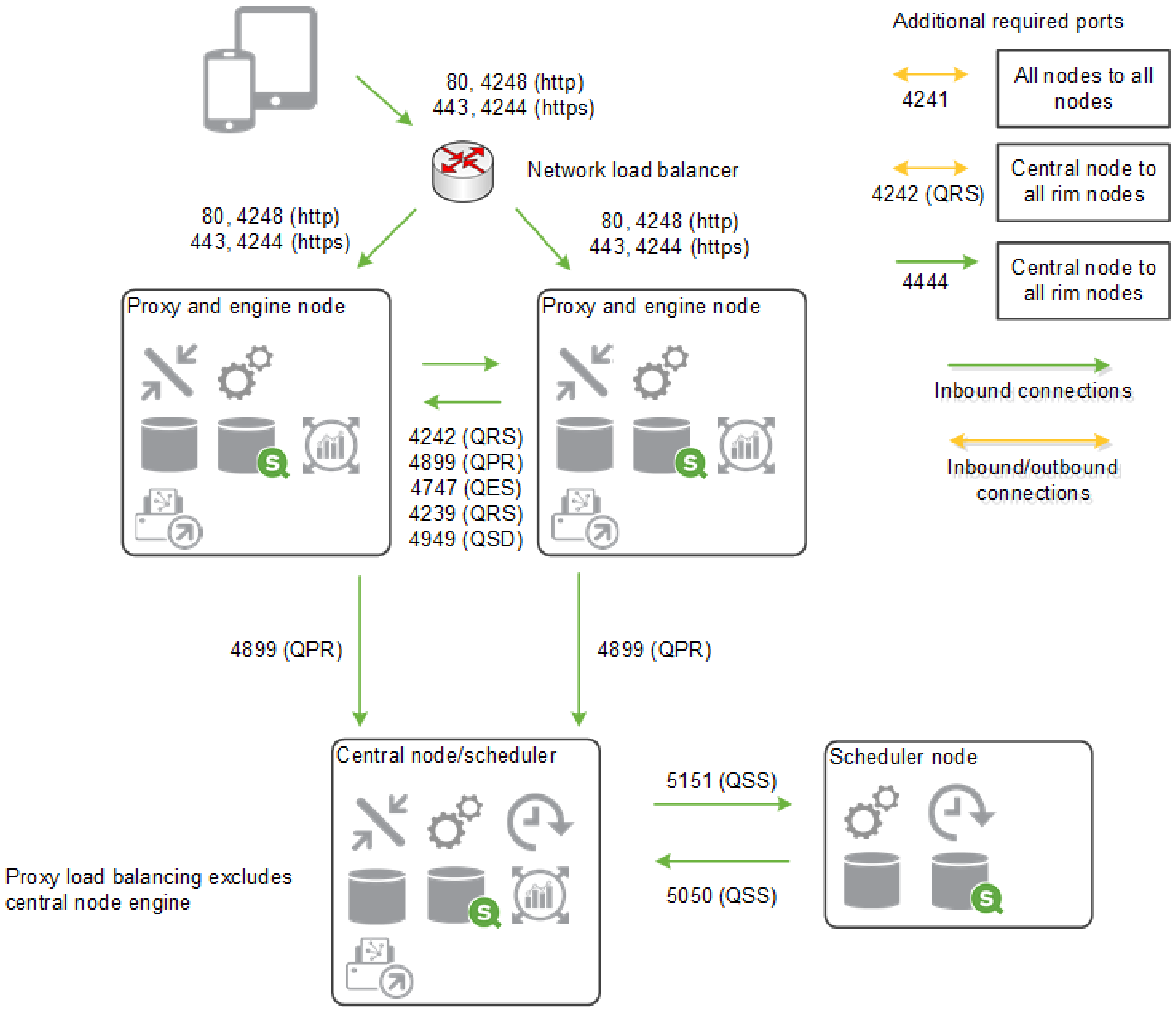
High availability proxy and engine nodes

This example shows the ports that are used in a multi-node site when deploying more than one proxy and engine node.



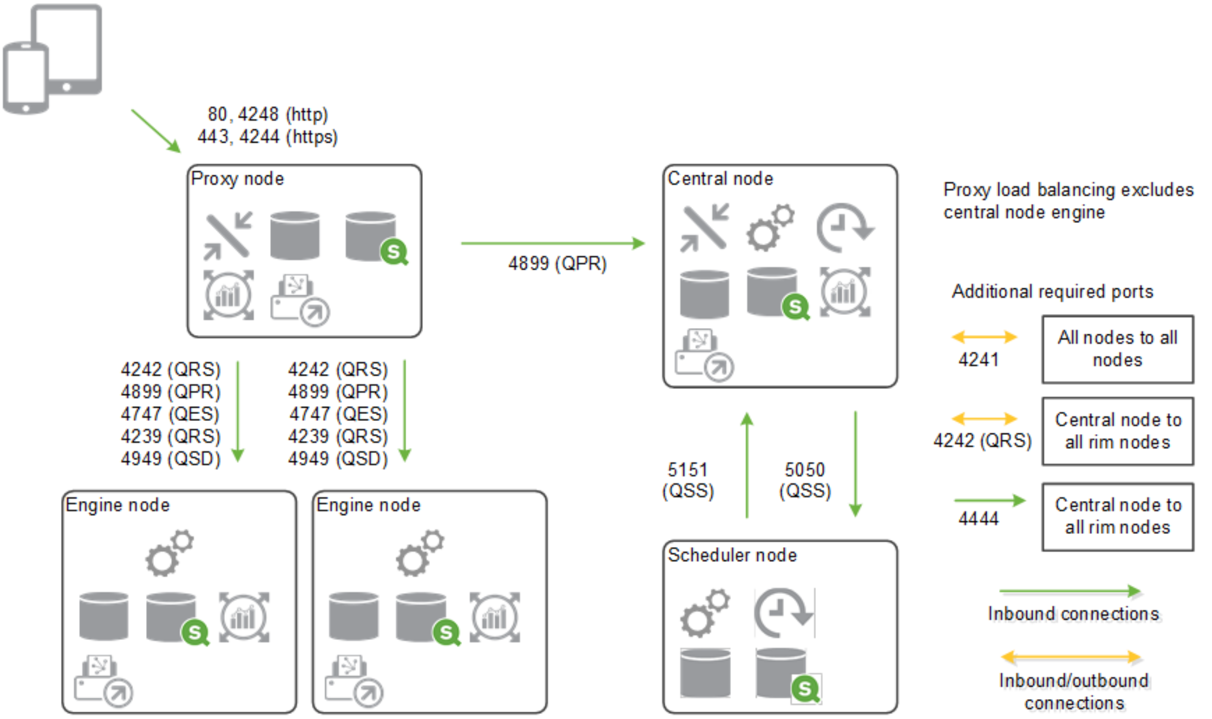
Separate scheduler node and high availability proxy and engine nodes

This example shows the ports that are used in a multi-node site when deploying a separate scheduler node and more than one proxy and engine node.



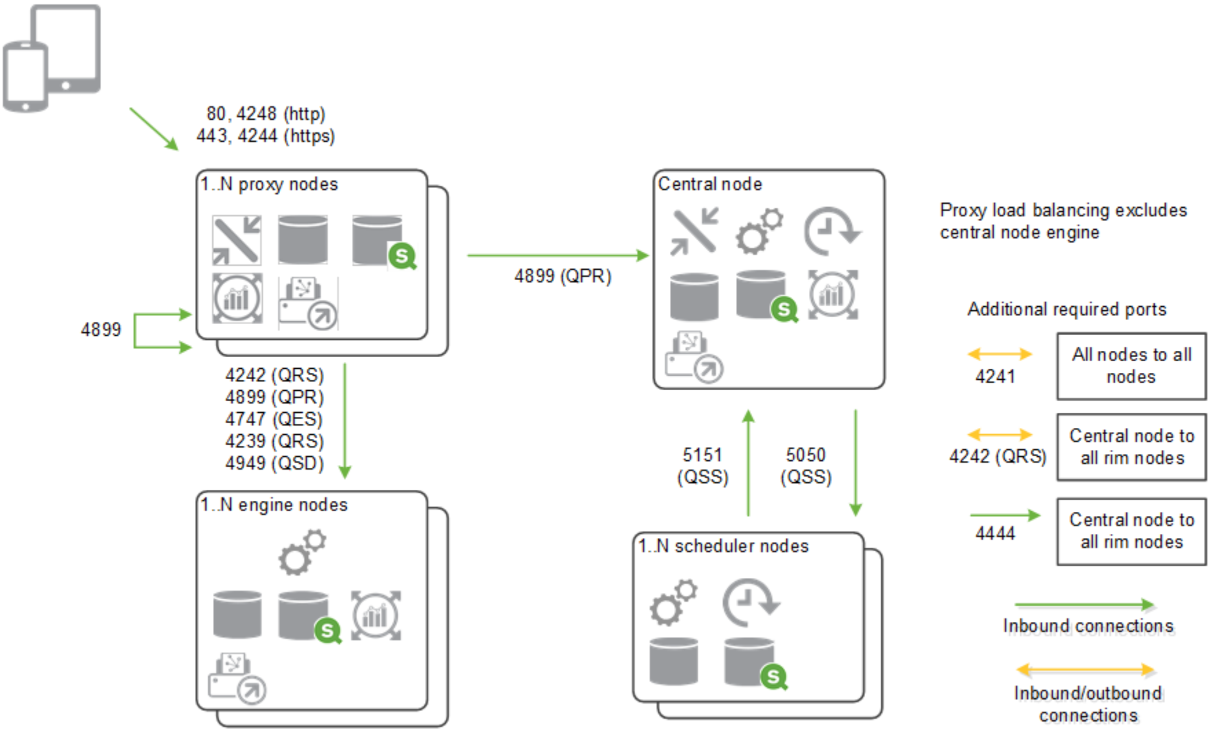
Separate proxy and scheduler nodes and high availability engine nodes

This example shows the ports that are used in a multi-node site when deploying separate proxy and scheduler nodes and more than one engine node.



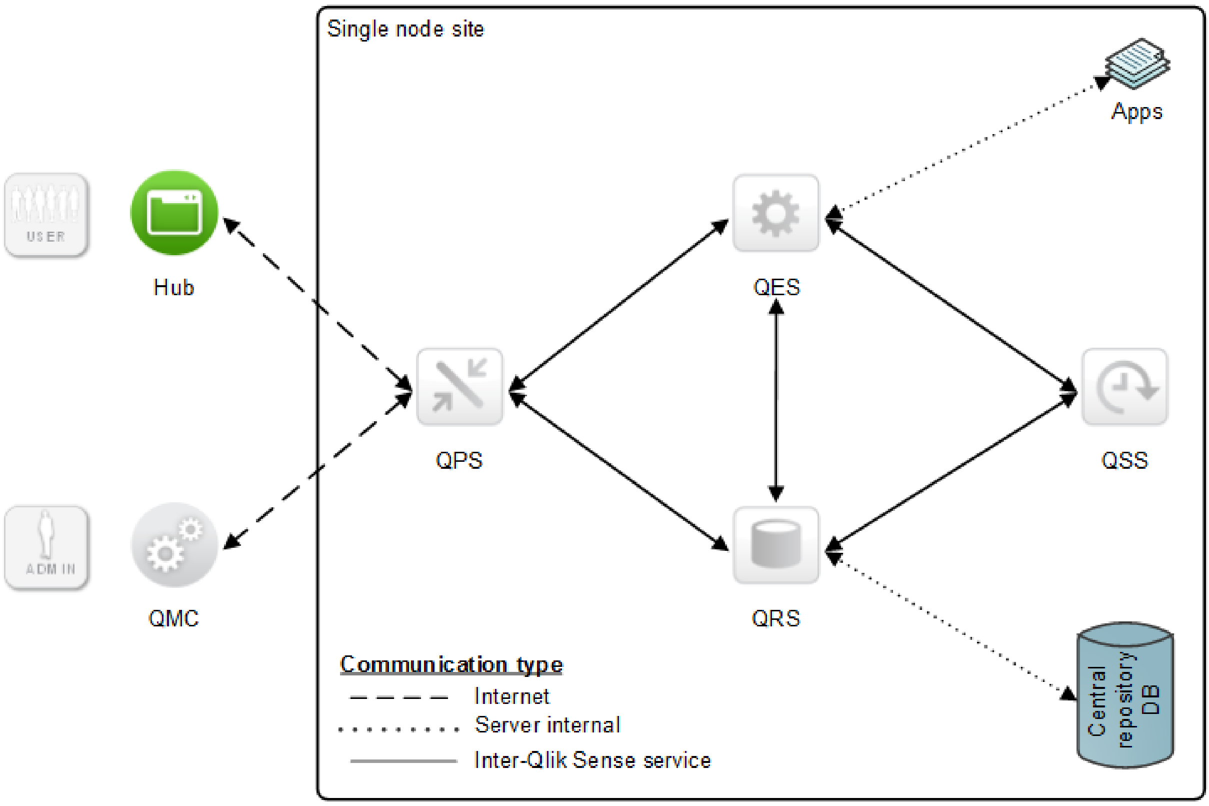
Generic scale out

This example shows the ports that are used in a multi-node site when scaling the site by adding additional proxy, engine, or scheduler nodes.



2 Deploying single node sites

In this deployment scenario, all Qlik Sense services run on a single node. This kind of deployment works best in a single time zone, where reloads of data can be done during the night.



See: *Architecture (page 12)*

2.9 Services

This section describes how the Qlik Sense services behave when deployed in single node sites.

Qlik Sense Repository Service

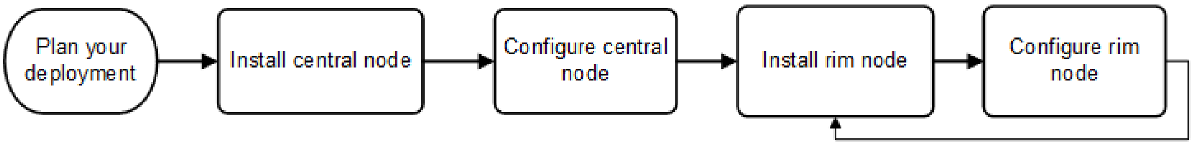
Within a single node site, there is only one instance of the Qlik Sense Repository Service (QRS) running and it has direct access to the central repository database.

Qlik Sense Scheduler Service

When deployed in a single node site, the Qlik Sense Scheduler Service (QSS) acts as both master and slave.

2 Deploying multi-node sites

This section provides guidance on what to consider when planning and designing multi-node sites.



This picture shows the general procedure steps for deploying a multi-node site. The individual steps will be different depending on the persistence model and topology of your deployment.

2.10 Planning your deployment

When planning the deployment of a multi-node site the areas described in this section need to be investigated.

Select persistence model

You can select to deploy repository and app storage in two different ways:

- Synchronized persistence
  In a synchronized persistence multi-node site, each node has a local copy of the data that it needs to fulfill its role, which means that the node can operate independently in the event of a server or network failure. Each node can read and write its local data and a synchronization mechanism in Qlik Sense distributes the changes to other nodes in the site.
- Shared persistence
  In a multi-node site with shared persistence nodes share a single repository database and a network folder for the application files. You can either share the central node repository and application storage, or setup dedicated machines for repository and application storage.

Amount of content to synchronize (synchronized persistence only)

Each node that serves an app needs a copy of the entire data model locally before it can allow users access. Each time the app is reloaded the changed data is synchronized.

This means that the factors that affect the total volume to synchronize are:

- The number of apps
- The size of the apps
- The reload frequency of the apps (and spread over time)

Multiple apps can be synchronized simultaneously. If an app is reloaded faster than it can be synchronized to the other nodes, the synchronization is canceled and starts over.

Based on the information above, it is possible to calculate the amount of data that needs to be synchronized each hour. This has to be compared to the amount of data that can be moved each hour, which is affected by factors such as network speed, the number of nodes, and the Qlik Sense software itself.

QVD files are not included in the synchronization and should not be considered as part of the total.

Number of nodes

The number of nodes needed in a multi-node site depends on whether you want to spread the load over fewer, but larger, servers or over more, but smaller, servers. Because of the synchronization in Qlik Sense using fewer, but larger, servers requires less synchronization traffic. The role of each server also needs to be considered.

Deployment topology

Qlik Sense is flexible when it comes to how the nodes are laid out and supports different needs for scale, security, geography, and resilience. In addition, the synchronization rules can be used to configure the role of each node and the content that it serves to the users. The requirements for this should be identified when planning the topology of the deployment. Shared persistence does not support geographical deployments.

2.11 Recommended deployment principles

It is recommended to consider the principles described in this section when planning the deployment of a multi-node site.

Dedicated roles per server

It is recommended to give each node a specific role within the deployment as this allows for better planning of resources and ensures consistent performance for the users. For example, specify if a node is to run scheduled reloads or serve content to users.

Dedicated hardware for the central node (large deployments only)

In large multi-node sites, the central node takes on the load of distributing content to the rim nodes. Depending on the usage, it may be good to dedicate resources for the central node, so that it does not have to compete for resources with user or reload traffic. The server used for the central node could also be a candidate for virtualization.

App development

Apps can be developed using Qlik Sense Desktop or in a Qlik Sense server environment. The latter provides advantages in terms of data governance, security, performance, and collaboration.

The development of apps involves building load scripts and running reloads while building the data model and assembling the user interface. As each iteration of changes made to an app is synchronized to the other nodes, a significant load can be generated on the synchronization process, especially if the app is large or reloaded often. It is therefore recommended to carefully choose the nodes used for development and avoid synchronization of content that does not need to be processed until the app is finalized.

There are two ways to handle app development in a multi-node site:

- Isolate the development activities onto a dedicated development node within a production site.
- Specify a dedicated site for development activities. The advantage of this approach is that it allows an organization to follow their preferred process for testing and releasing new apps.

If the app development is especially intensive, it is recommended to use a dedicated development site.

It is primarily the reload of apps in development that needs to be considered. Adding the user interface elements only has a minor effect on the synchronization of content.

Synchronize only what is needed (synchronized persistence only)

Synchronization rules can be used to significantly reduce the amount of data traffic. For example, an unpublished app that is only used by a developer does not have to be synchronized to nodes that only serve published apps to end users. In addition, apps that only reload QVD files do not need to be synchronized.

Network speed and geographic deployments

The ability to move app content in a reasonable time is affected by the network speed. The better the throughput, the faster the synchronization can be. For best results, the nodes should be on the same network or, if in separate data centers, connected by LAN like network connections.

If the nodes are geographically dispersed and on slower networks, the synchronization will be slower. This may also slow down the synchronization of nodes that are on faster connections. In this case, the volume and frequency of the data to synchronize need to be considered.

If the data is changed faster than the network or software can synchronize it to other nodes (for example, by reloading again before the synchronization completes), the synchronization is canceled. This may lead to nodes never being updated or synchronization queues that may affect the user experience.

Deployment size

The size (that is, the number of nodes) of a deployment depends of the following factors:

- The amount of data to move (per hour)
- The number of nodes that need to receive the data
- The network speed available to transfer the content
- Some overhead for the software and available CPU

It can be difficult to define the number of nodes or possible data transfer ability, but the following should be considered:

- Less data + fast network = You can have more nodes and the synchronization time will be short
- More nodes + slow network = Less data can be synchronized without delays

Examples:

- Moving 5 GB of content (for example, reloaded apps) or less in an hour between eight servers on a typical corporate network should result in quick updates to the content.
- Moving a single 12 GB app between ten nodes on a typical corporate network may take up to 30 minutes. This means that apps of this size should not reload frequently.

2.12 Deploying multi-node sites with shared persistence

In a multi-node site, the site is spread out across two or more nodes that share the same set of data and license key. nodes share a single repository database and a network folder for the application files. You can either share the central node repository and application storage, or setup dedicated machines for repository and application storage.

See: *Architecture (page 12)*

Planning a multi-node site with shared persistence

Qlik Sense supports two types of shared persistence deployments:

- Database and file storage are hosted on one of the nodes in the site. This type of deployment requires no extra installation or additional hardware.
- Database and file storage are hosted on one or more dedicated machines. This type of deployment requires one or more additional servers for running a PostgreSQL cluster and for hosting file shares, but provides more options for high availability deployments and more flexibility in terms of server deployments.

The functionality is the same with both options, and the set up steps are very similar, with a few additional steps required when using a dedicated machine. It is also possible to deploy the database on a separate machine, and use one of the nodes to store files.

If the deployment is small, such as 2-3 nodes, or there are no specific high availability requirements, we recommend using the database and file storage on one of the server machines. In larger deployments we recommend that you separate the database and file storage, which in addition will provide high availability options.

If you are installing custom connectors in a multi-node setup, the custom connectors must be installed on each node.

Limitations

There are some limitations to shared persistence deployments.

Geographical deployments Shared persistence does not support geographical deployments, as Qlik Sense servers need to be located in the same local network as the database and the file share.

Capacity and performance Shared persistence supports up to a maximum of 8 machines. In addition to the number of nodes, there are other factors that will contribute to total capacity:

- workload
- hardware speed
- network speed

For example, if the disk speed of the file share and the central node is too slow, you may expect low performance during some operations, such as importing or duplicating apps.

We recommend scalability testing and engaging with Qlik Consulting for large deployments.

DMZ deployments All machines in a site, including machines without an engine, require access to both the database and file share. In DMZ deployments this may require opening additional ports, or taking an alternative approach, compared to a DMZ deployment with synchronized persistence.

Central node dependencies The central node is responsible for handling a number of vital operations in the site. When the central node is put offline, those operations are prevented from running, including:

- master scheduler, responsible for triggering reloads
- allowing new users to obtain a license
- using extension objects

Security considerations

In shared persistence deployments the network traffic between the servers, the database and the file share is not encrypted by default after an installation. We recommend applying the settings described in the following sections:

- Setting up database traffic encryption
- Optional: Setting up database replication and failover

Deploying a site with the central node hosting database and files

The basic steps for deploying a multi-node site with the central node hosting database and files are as follows:

1. Plan the nodes that are needed in the deployment.
2. Install the first node in the site and create a new cluster. This node becomes the central node, which contains the database and all files that are needed.
   See: Install and upgrade Qlik Sense
3. Install an additional node and add it to the cluster.

See: Install and upgrade Qlik Sense

4. In the Qlik Management Console (QMC) on the central node, add the new node to the site.

   See: Manage Qlik Sense sites

5. Test the new node.
6. Continue to install additional nodes – one at a time – as described in step 3 until the multi-node site is complete.

Deploying a site with database and file storage hosted on separate machines

The basic steps for deploying a multi-node site with the central node hosting database and files are as follows:

1. Plan the nodes that are needed in the deployment.
2. Install and configure the database server.
3. Set up a file share to store files.
4. Install the first node in the cluster. This node becomes the central node.

   See: Install and upgrade Qlik Sense

5. Install an additional node and add it to the cluster.

   See: Install and upgrade Qlik Sense

6. In the Qlik Management Console (QMC) on the central node, add the new node to the site.

   See: Manage Qlik Sense sites

7. Test the new node.
8. Continue to install additional nodes – one at a time – as described in step 3 until the multi-node site is complete.

2.13 Deploying multi-node sites with synchronized persistence

In a multi-node site, the site is spread out across two or more nodes that share the same set of data and license key. In a synchronized persistence multi-node site, each node has a local copy of the data that it needs to fulfill its role, which means that the node can operate independently in the event of a server or network failure. Each node can read and write its local data and a synchronization mechanism in Qlik Sense distributes the changes to other nodes in the site.

The basic steps for deploying a multi-node site with synchronized persistence are as follows:

1. Plan the nodes that are needed in the deployment.
2. Install the first node in the site. This node becomes the central node, which contains all apps that are needed.

   See: Install and upgrade Qlik Sense

3. Install an additional node as a rim node.

   See: Install and upgrade Qlik Sense

4. In the Qlik Management Console (QMC) on the central node, add the new node to the site.

   See: Manage Qlik Sense sites

5. Wait for the first synchronization to complete, and then test the new node.
6. Continue to install additional rim nodes – one at a time – as described in step 3 until the multi-node site is complete.

If you are installing custom connectors in a multi-node setup, the custom connectors must be installed on each node.

2.14 Services in multi-node site deployments

This section describes how the Qlik Sense services behave when deployed in multi-node sites.

Qlik Sense Repository Service

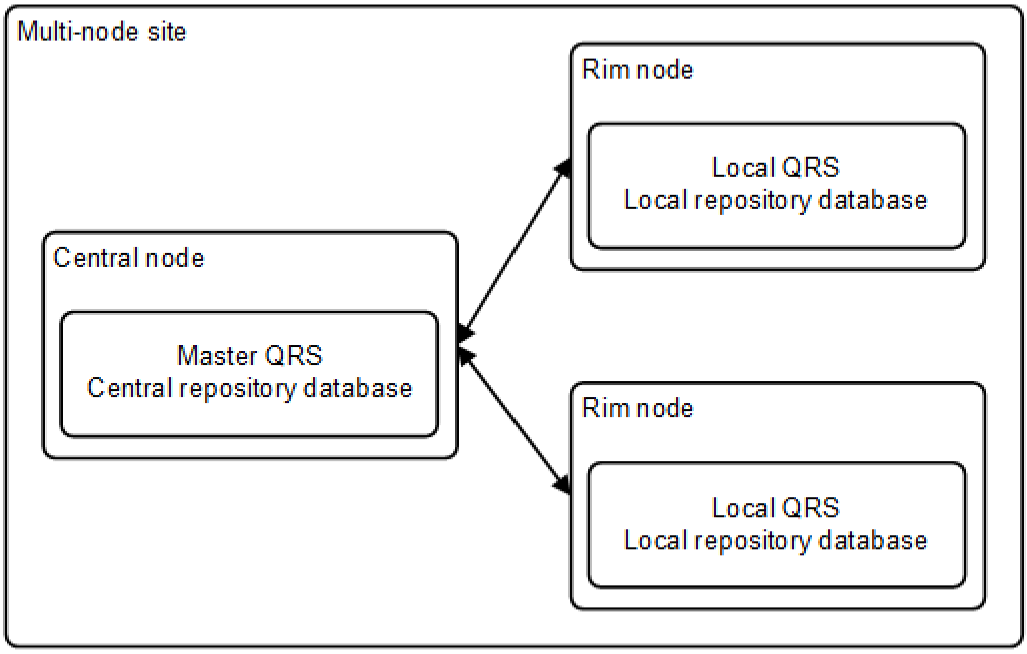
The Qlik Sense Repository Service (QRS) behaves differently depending on if it is deployed on the central node or on a rim node in a site with synchronized persistence, or if the site is deployed with shared persistence.

All nodes in a site with shared persistence

Within a multi-node site with shared persistence, one instance of the Qlik Sense Repository Service (QRS) runs on each node, and all nodes have direct access to the shared repository database.

Central node in a site with synchronized persistence

Within a multi-node site with synchronized persistence, one instance of the Qlik Sense Repository Service (QRS) runs on each node. The QRS running on the central node is considered to be the master. The master QRS has direct access to the central repository database, whereas the other QRSs only have access to a local repository database on the node where they are running. The master QRS synchronizes the central repository database and the local repository databases.


Multi-node site
Central node
Master QRS
Central repository database
Rim node
Local QRS
Local repository database
Rim node
Local QRS
Local repository database

When the master QRS starts, it connects to the central repository database. If no database exists, the master QRS builds the database and populates it with initial data. In a default Qlik Sense installation, the repository database is a specific instance of PostgreSQL that runs its own database cluster specifically for the repository.

Rim nodes in a site with synchronized persistence

When the Qlik Sense Repository Service (QRS) on a rim node starts, it connects to the local repository database on the node. If no local repository database exists, the QRS waits until it communicates with the central node.

In a default Qlik Sense installation, the repository database is a specific instance of PostgreSQL that runs its own database cluster specifically for the repository.

Qlik Sense Proxy Service

On the central node in a multi-node site, it is recommended to have a dedicated Qlik Sense Proxy Service (QPS) that is used specifically for the Qlik Management Console (QMC) and not for the hub.

Qlik Sense Scheduler Service

Depending on the type of deployment, the Qlik Sense Scheduler Service (QSS) runs as master, slave, or both on a node.

Master

There is only one master Qlik Sense Scheduler Service (QSS) within a site and it is always located on the central node, where the master Qlik Sense Repository Service (QRS) runs. This means that the central node must have the Qlik Sense Scheduler Service (QSS) installed even if more QSS nodes are added. This is because the QSS on the central node coordinates all QSS activities within the site.

The master QSS handles all task administration (for example, which tasks to execute and when to execute a specific task). When the time comes to execute a task, the master QSS sends the task ID to a slave QSS within the site. Which slave QSS to distribute the task ID to is determined by a load balancing operation performed by the master QSS.

When a slave QSS completes a task, it returns the task state (successful or fail) to the master QSS. The master QSS uses the task state to perform task chaining, which means that it uses the task state to determine if other events are affected by the state of the completed task and need to be executed. This can be configured in the Qlik Management Console (QMC).

If the slave QSS fails to perform the task, the master QSS repeatedly requests the same or another slave QSS to perform the task until it has been completed or until the maximum number of attempts has been reached.

Slave

If a Qlik Sense Scheduler Service (QSS) runs on a rim node, the QSS is considered to be a slave QSS.

When receiving a task ID from the master QSS, the slave QSS reads the task from the local repository database and executes the task.

When a slave QSS completes a task, it returns the task state (successful or fail) to the master QSS.

Qlik Sense Engine Service

On the central node in a multi-node site, it is recommended to have a dedicated Qlik Sense Engine Service (QES) that is used specifically for the Qlik Management Console (QMC) and not for the hub.

Qlik Sense Printing Service

Within a multi-node site, one instance of the Qlik Sense Printing Service (QPR) runs on each node.

Export requests from clients are directed to the printing services in the multi-node site using round robin load balancing. For example, if the first export request is load balanced to the QPR on node 1, the second export request is load balanced to the QPR on node 2, and so on.

Qlik Sense Service Dispatcher

Broker Service

The Broker Service acts as an interface to and an intermediary between services started by the Qlik Sense Service Dispatcher (QSD).

The service is launched and managed by the Qlik Sense Service Dispatcher (QSD) when required.

Data Profiling Service

The Data Profiling Service communicates directly with the Qlik Sense Engine Service (QES) on the node.

The service is launched and managed by the Qlik Sense Service Dispatcher (QSD) when required.

Hub Service

The Hub Service controls which content a user is allowed to see based on authorization.

The service is launched and managed by the Qlik Sense Service Dispatcher (QSD) when required.

Migration Service

The Migration Service only runs on the central node in a site.

The service is launched and managed by the Qlik Sense Service Dispatcher (QSD) when required.

2.15 Example deployment scenarios

This section provides examples of how multi-node sites can be deployed. All examples are based on a site with synchronized persistence.

The following terms are used in the scenarios:

- Central node: The node that is responsible for management activities and synchronization.
- Reload/scheduler node: A node that reloads apps on a schedule, but serves no content to users.
- Consume node: A node that serves apps to users, but is not used to create, process, or reload data.
- Development node: A node that allows users to create and reload new apps, but does not serve normal consumer traffic.
- Proxy node: A node that provides load balancing of user traffic to other nodes. This node does not contain a Qlik Sense Engine Service (QES).

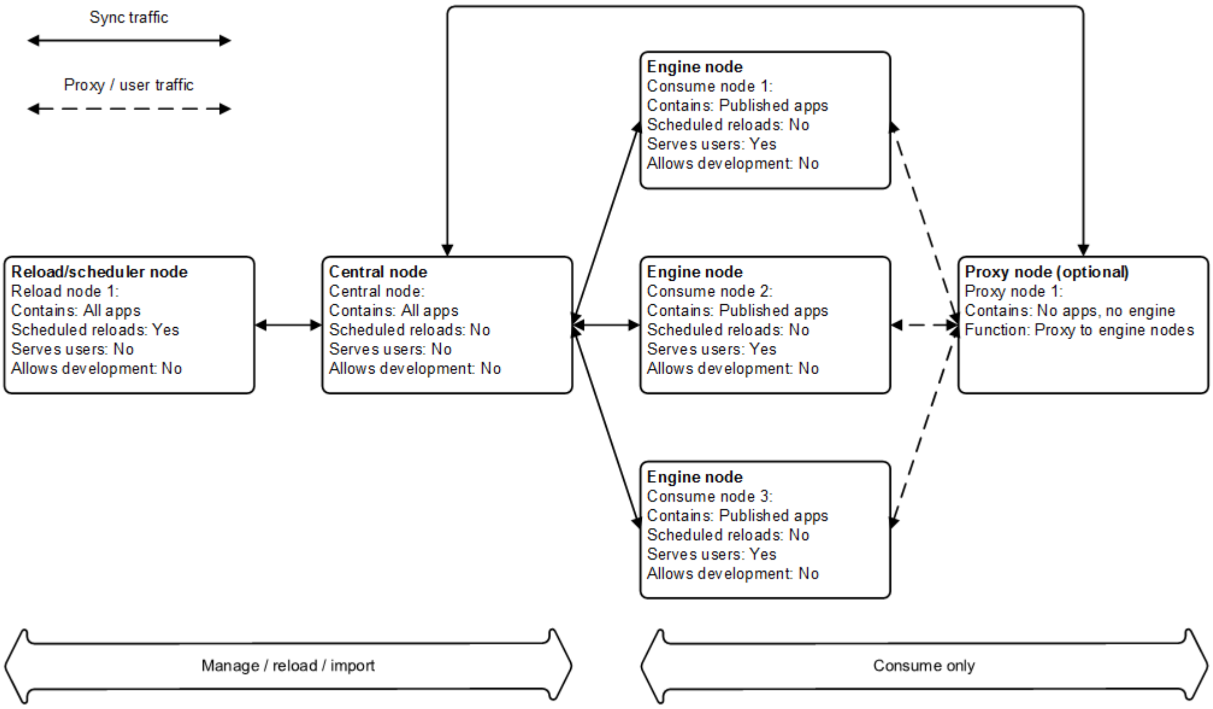
An alternative to use a proxy node is to have a proxy installed on each consume node and balance the traffic using a hardware load balancer.

Multi-node scenario: Production deployment

This scenario describes how to setup a typical internal production deployment, which provides the ability to scale up both reloads and user load.

Node layout

Each node within the site only contains the services and data that it needs to perform its role.



Services on each node

The table below lists the Qlik Sense services that are deployed on each node in the site.

Node name	Qlik Sense Repository Service (mandatory)	Qlik Sense Engine Service	Qlik Sense Scheduler Service	Qlik Sense Proxy Service
Central node	x	x	x	x
Reload/scheduler node(s)	x	x	x	
Consume node (s)	x	x		
Proxy node	x			x

The table does not list Qlik Sense services that are deployed automatically on nodes, for example, the Qlik Sense Printing Service (QPR) and the Qlik Sense Service Dispatcher (QSD).

Configuration steps

Basic installation

Proceed as follows to perform the basic installation:

1. Starting with the central node, install the Qlik Sense software and services as described in the table above.

   See: Install and upgrade Qlik Sense

2. Add each rim node via the Qlik Management Console (QMC) on the central node.

   See: Manage Qlik Sense sites

   > ⓘ *Specify the consume nodes to be **Production** nodes.*

   > ⚠ *The DNS names used must be correct.*

3. When all rim nodes have been added, check that they are displayed as being online in the QMC on the central node.

Load balancing

Proceed as follows to configure the load balancing:

1. Select **Virtual proxies** in the Qlik Management Console on the central node.
2. Edit the settings for the proxy node.

   Under **Load balancing nodes**, specify that the consume nodes should be used.

3. Check that the hub is accessible on the proxy node. In addition, check that the hub lists the apps.

Qlik Sense Scheduler Service

Proceed as follows to configure the Qlik Sense Scheduler Service (QSS):

- Configure the QSS on the central node to run as master only (that is, do not run reloads on the central node). The reload node should be set as slave, which means it will handle all reloads.

License Monitor and Operations Monitor

In multi-node environments, the License monitor and Operations monitor apps can be reloaded on any node.

Proceed as follows:

1. Share the Qlik Sense folder on the central node.

   The default location is *%ProgramData%\Qlik\Sense*.

2. Update the data connections *ServerLogFolder* and *ArchivedLogFolder* by replacing *%ProgramData%\Qlik\Sense* with the path to the shared folder.
3. Reload the apps on the rim nodes.

Because of the way multi-node environments are synchronized and the way logs are archived, the results of reloads may not be completely current. Reloads include all logs from the *ArchivedLogFolder* folder on the central node and the active *.txt* log files stored in the *Sense\Log* folder on the central node.

Multi-node scenario: Production deployment allowing development

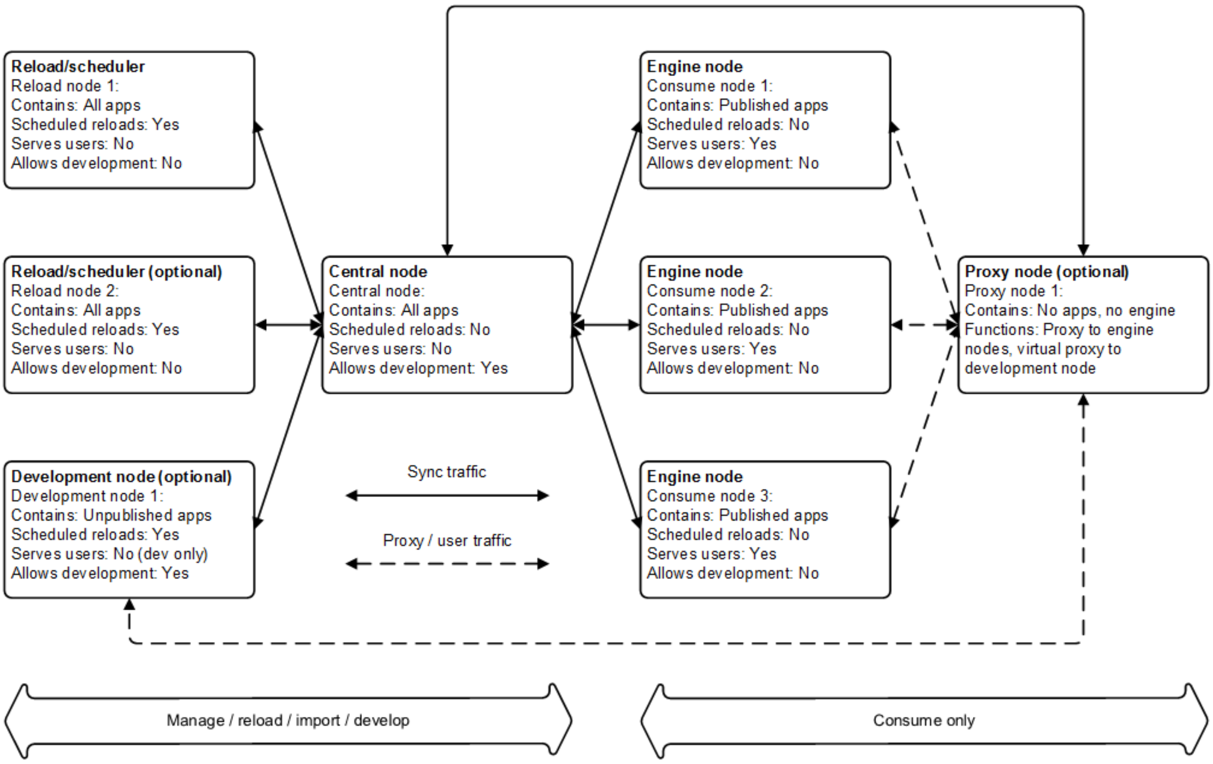
This scenario describes how to setup a typical internal production deployment that allows app development on dedicated nodes and uses synchronization rules to reduce the synchronization traffic.

Both reloads and user load can be scaled up. Additional developer, reload, consume, and proxy nodes can be added as long as the total number is in line with the recommendations.

The synchronization rules described in this section are just examples. The exact approach depends on the deployment needs.

Node layout

Each node within the site only contains the services and data that it needs to perform its role.



Services on each node

The table below lists the Qlik Sense services that are deployed on each node in the site.

Node name	Qlik Sense Repository Service (mandatory)	Qlik Sense Engine Service	Qlik Sense Scheduler Service	Qlik Sense Proxy Service
Central node	x	x	x	x

Node name	Qlik Sense Repository Service (mandatory)	Qlik Sense Engine Service	Qlik Sense Scheduler Service	Qlik Sense Proxy Service
Reload/scheduler node(s)	x	x	x	
Consume node (s)	x	x		
Proxy node	x			x
Development node	x	x		

The table does not list Qlik Sense services that are deployed automatically on nodes, for example, the Qlik Sense Printing Service (QPR) and the Qlik Sense Service Dispatcher (QSD).

Configuration steps

Basic installation

Proceed as follows to perform the basic installation:

1. Starting with the central node, install the Qlik Sense software and services as described in the table above.

   See: Install and upgrade Qlik Sense

2. Add each rim node via the Qlik Management Console (QMC) on the central node.

   See: Manage Qlik Sense sites

   *Specify the consume nodes to be **Production** nodes and the development node to be a **Development** node.*

   The DNS names used must be correct.

3. When all rim nodes have been added, check that they are displayed as being online in the QMC on the central node.

Load balancing: Consume nodes

Proceed as follows to configure the load balancing:

1. Select **Virtual proxies** in the Qlik Management Console on the central node.
2. Edit the settings for the proxy node.

Under **Load balancing nodes**, specify that the consume nodes should be used.

3. Check that the hub is accessible on the proxy node. In addition, check that the hub lists the apps.

Load balancing: Development node

In this example, a virtual proxy is used to route traffic to the development node. The means that a slightly different URL, *https://<server address>/DEV/hub*, is used access the development section of the site. The *DEV* prefix also means that the development traffic can be handled differently compared to the normal user traffic.

Proceed as follows to configure the load balancing:

1. Select **Proxies** in the Qlik Management Console (QMC) on the central node.
2. Edit the settings for the proxy node.
3. Select **Virtual proxies**, click **Add**, and then select **Create new**.
4. Provide a description and the *DEV* prefix. As cookie names have to be unique, add *DEV* to the end of the session cookie field.
5. Under **Load balancing nodes**, specify that the development node should be used.
6. Check that the development node is available using the following URL:

   https://<server address>/DEV/hub

Configuring custom properties

Custom properties can be used in the synchronization rules to dictate the nodes to which to synchronize apps.

Proceed as follows to configure the custom properties:

1. Select **Custom properties** in the Qlik Management Console (QMC).
2. Add the following custom property:
   - **Name**: *NodeType*
   - **Resource types**: *Nodes*
   - **Values**: *Dev*, *Consume*, *Reload*
3. Select **Nodes** in the QMC.
4. For each node in the site, set the appropriate value for *NodeType* under **Custom properties**.

Configuring synchronization rules

The synchronization rules are used to synchronize the apps to the right nodes.

Proceed as follows to disable the default synchronization rule, which allows all apps to be synchronized to all nodes:

1. Select **Sync rules** in the Qlik Management Console (QMC).
2. Disable the default sync rule, ResourcesOnNonCentralNodes.

   After a few seconds, all apps disappear from the proxy node as they are only available on the central node.

New rules are needed to synchronize the apps to the correct nodes and prevent unnecessary synchronization. For example, even though users cannot see an unpublished app (that is, an app under development) until it is published, the unpublished copy is by default synchronized to all nodes. In this example, rules are setup to stop unpublished apps from being synchronized to the nodes used by consumer users and to ensure that the reload nodes only get what they need for their workload. Ideally, the apps should be distributed as follows:

- Reload nodes: Apps to reload
- Consume nodes: Published apps
- Development node: Unpublished apps

Proceed as follows to configure the new synchronization rules:

1. Create a rule to synchronize unpublished apps to the development node:
   1. Create a new synchronization rule in the QMC.
   2. Name the new rule *unpublished apps to dev nodes* and then select the **Advanced** option.
   3. Enter the following in the **Conditions** field and then save the rule:

      (node.@NodeType="Dev" and resource.stream.name.Empty())
      The condition states that if the app is not in a stream (and thus is not yet published), synchronize it to a node with the *NodeType* set to *Dev*.
   4. Browse to the development node proxy and check the results.
2. Create a rule to synchronize published apps to the consume nodes:
   1. Create a new synchronization rule in the QMC.
   2. Name the new rule *published apps to consumer nodes* and then select the **Advanced** option.
   3. Enter the following in the **Conditions** field and then save the rule:

      (node.@NodeType="Consume" and !resource.stream.name.Empty())
      The condition states that if the stream value is not empty (meaning that the app is published), synchronize the app to a node with the *NodeType* set to *Consume*.
   4. Browse to the main proxy server URL and check the results.
3. Create a rule to synchronize apps to the reload nodes:
   1. Create a new synchronization rule in the QMC.
   2. Name the new rule *apps to reload nodes* and then select the **Advanced** option.
   3. Enter the following in the **Conditions** field and then save the rule:

      (node.@NodeType="Reload")
      The condition states that if the *NodeType* is set to *Reload*, all apps are synchronized to the node.
      To limit the traffic to these nodes, apply custom properties to the apps that should be synchronized to the nodes. For example, create a custom property called *apptype* and use a rule like the following:
      (node.@NodeType="Reload" and (resource.@AppType="QVDLoader" or !resource.stream.name.Empty()))
      This rule synchronizes all published apps and apps tagged as QVD loaders (that is, all the apps to run scheduled reloads on) to the reload nodes.

Qlik Sense Scheduler Service

Proceed as follows to configure the Qlik Sense Scheduler Service (QSS):

- Configure the QSS on the central node to run as master only (that is, do not run reloads on the central node). The reload node should be set as slave, which means it will handle all reloads.

License Monitor and Operations Monitor

In multi-node environments, the License monitor and Operations monitor apps can be reloaded on any node.

Proceed as follows:

1. Share the Qlik Sense folder on the central node.

   The default location is *%ProgramData%\Qlik\Sense*.
2. Update the data connections *ServerLogFolder* and *ArchivedLogFolder* by replacing *%ProgramData%\Qlik\Sense* with the path to the shared folder.
3. Reload the apps on the rim nodes.

Because of the way multi-node environments are synchronized and the way logs are archived, the results of reloads may not be completely current. Reloads include all logs from the *ArchivedLogFolder* folder on the central node and the active *.txt* log files stored in the *Sense\Log* folder on the central node.

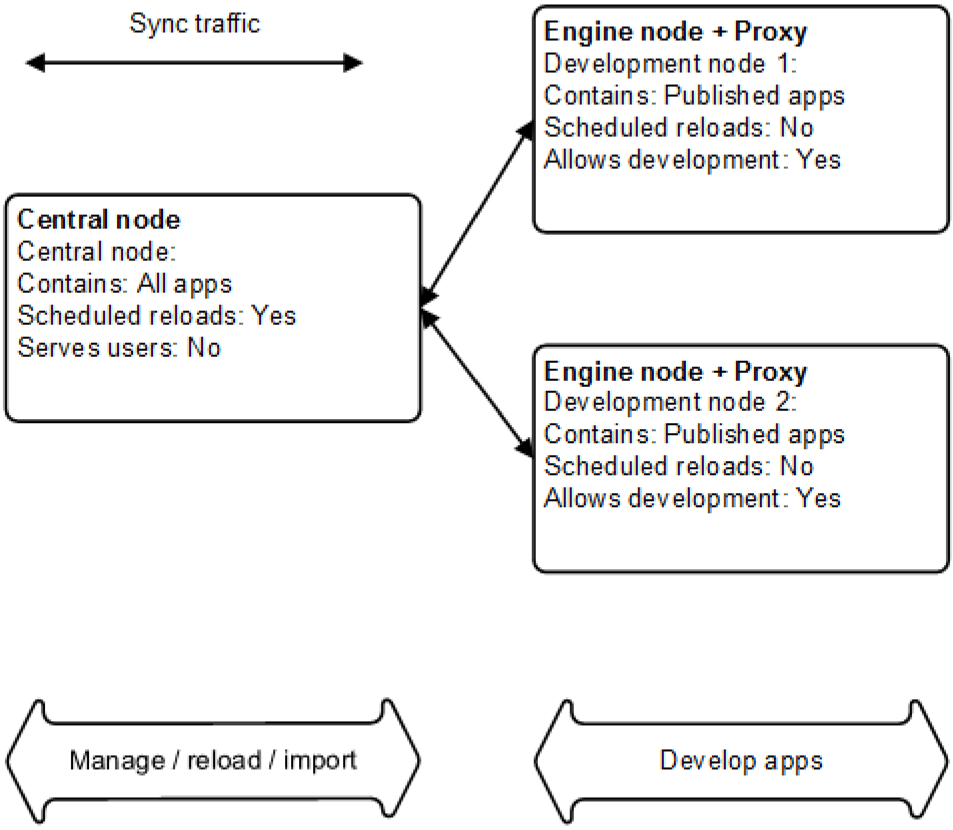
Multi-node scenario: Development site

This scenario places the development of apps onto dedicated resources. The number of nodes can be adjusted to support the amount of development activity (for example, a single node can be used).

If more than one development node is used, they can be load balanced using a proxy node. However, when creating a new app there can be a short delay before the app is added on all nodes, which means that the users may be routed to a node that has not yet received the new app.

Node layout

Each node within the site only contains the services and data that it needs to perform its role.



Services on each node

The table below lists the Qlik Sense services that are deployed on each node in the site.

Node name	**Qlik Sense Repository Service (mandatory)**	**Qlik Sense Engine Service**	**Qlik Sense Scheduler Service**	**Qlik Sense Proxy Service**
Central node	x	x	x	x
Development node(s)	x	x		x

The table does not list Qlik Sense services that are deployed automatically on nodes, for example, the Qlik Sense Printing Service (QPR) and the Qlik Sense Service Dispatcher (QSD).

Configuration steps

Basic installation

Proceed as follows to perform the basic installation:

1. Starting with the central node, install the Qlik Sense software and services as described in the table above.

   See: Install and upgrade Qlik Sense

2. Add each rim node via the Qlik Management Console (QMC) on the central node.

   See: Manage Qlik Sense sites

   > ⚠ *The DNS names used must be correct.*

3. When all rim nodes have been added, check that they are displayed as being online in the QMC on the central node.

Load balancing

Proceed as follows to configure the load balancing:

1. Select **Virtual proxies** in the Qlik Management Console on the central node.
2. Edit the settings for the proxy node.

   Under **Load balancing nodes**, specify that the local Qlik Sense Engine Service (QES) should be used for each proxy.

3. Check that the hub is accessible on the development nodes. In addition, check that the hub lists the apps on the nodes and that new apps can be created.

3 Backing up and restoring

This section describes how to back up and restore Qlik Sense sites and certificates and how to move a node to a new machine.

3.1 Backing up and restoring a site

This section describes how to backup and restore a Qlik Sense site. This can be done manually or using the Repository Snapshot Manager (RSM):

- Manually: See *Backing up a site manually (page 59)* and *Restoring a site manually (page 62)*
- Using the RSM: See *Using the Repository Snapshot Manager (page 66)*

> *These instructions define the minimum steps required. The use of specific backup software may further extend the options for backup and restore.*

In a single node site, the single node is referred to as the central node.

In a multi-node site, the central node is the master record that contains all data about the site. The rim nodes in the multi-node site contain either a full copy or a limited subset of the data, which is maintained by the synchronization mechanism. This means that the central node is the only node that needs to be backed up in order to keep the data and configuration safe. The rim nodes can be restored by simply re-adding them as new nodes after a clean installation, since they will have their data restored by the synchronization mechanism.

Rim nodes maintain local log files that may be worth backing up in order to identify and investigate issues. It may also be worth backing up any general operating system data that may be required.

See also:

Architecture (page 12)

Backing up a site manually

This section describes how to backup a Qlik Sense site in a default installation where a PostgreSQL database is used as the repository database.

> *The instructions in this section are only applicable to the central node of a Qlik Sense site.*
>
> *See: Backing up and restoring a site (page 59)*

Items to backup

The following items need to be considered when backing up a site:

- Repository database: The database contains all configuration data for the site.
- Certificates for the Qlik Sense services: The certificates are used to encrypt the traffic between the services and the users. Make sure to backup the certificates in order not to lose any encrypted data (for example, passwords for data connections).
- Log data
- Application data: The data models in the Qlik Sense apps.
- Any content that supports the apps (for example, QVD files)

Backing up a site with shared persistence

Proceed as follows to backup a Qlik Sense site with shared persistence:

1. Make a backup of the certificates used to secure the Qlik Sense services. This only needs to be done once.

   See: *Backing up certificates (page 73)*

2. On all nodes of the site, stop all Qlik Sense services except the Qlik Sense Repository Database (QRD).
3. Make a backup of the repository database. In a shared persistence deployment with the database on a dedicated machine, this needs to be performed on the database server, using the installation location that you selected.

   > *The Repository Snapshot Manager cannot be used to backup sites deployed with shared persistence. The Repository Snapshot Manager is only supported for use with synchronized persistence deployments.*

   a. Open a Command Prompt with administrator privileges in Microsoft Windows.

   b. Produce a dumpfile for the repository database (that is, a single file for the entire database):

      i. Navigate to the installation location.
      Local database on the central node:

      ```
      cd "%ProgramFiles%\Qlik\Sense\Repository\PostgreSQL\<database version>\bin"
      ```

      Dedicated database server with default PostgreSQL install path:

      ```
      cd "%ProgramFiles%\PostgreSQL\9.3\data\bin"
      ```

      ii.

      ```
      pg_dump.exe -h localhost -p 4432 -U postgres -b -F t -f "c:\QSR_backup.tar" QSR
      ```

      If you are prompted for the PostgreSQL super user password, enter the password that was given during the installation of Qlik Sense.

      > *To avoid being prompted for the password (for example, if you want to automate the Qlik Sense backup process), you can use the pgpass functionality in PostgreSQL. See the PostgreSQL documentation for more information.*

      iii. Make a backup of the dumpfile for the repository database.

4. Make a backup of log and application data in the file share used for storage of log and application

data.

5. Make a backup of any locations where content that supports the Qlik Sense environment may be kept (for example, QVD files created by load scripts).
6. Start the Qlik Sense services. If the services are started manually, start them in the following order:
   a. Qlik Sense Repository Service (QRS)
   If the user running Qlik Sense services is not local administrator on the machine, you need to start Repository.exe from an elevated command prompt using the -bootstrap parameter.
   See: *Qlik Sense Repository Service (page 20)*

   b. Qlik Sense Proxy Service (QPS), Qlik Sense Engine Service (QES), Qlik Sense Scheduler Service (QSS), and Qlik Sense Printing Service (QPR) in no specific order

   The order is important because the QRS is dependent on the QRD and the rest of the services are dependent on the QRS.

Backing up a site with synchronized persistence

The procedure described in this section can also be performed using the Repository Snapshot Manager (RSM).

See: *Using the Repository Snapshot Manager (page 66)*

Proceed as follows to backup a Qlik Sense site:

1. Make a backup of the certificates used to secure the Qlik Sense services. This only needs to be done once.
   See: *Backing up certificates (page 73)*
2. Stop all Qlik Sense services except the Qlik Sense Repository Database (QRD).
3. Make a backup of the repository database.
   a. Open a Command Prompt with administrator privileges in Microsoft Windows.
   b. Produce a dumpfile for the repository database (that is, a single file for the entire database):
      i. Navigate to the installation location:
      cd "%ProgramFiles%\Qlik\Sense\Repository\PostgreSQL\<database version>\bin"
      ii. *pg_dump.exe -h localhost -p 4432 -U postgres -b -F t -f "c:\QSR_backup.tar" QSR*
      If you are prompted for the PostgreSQL super user password, enter the password that was given during the installation of Qlik Sense.

      > *To avoid being prompted for the password (for example, if you want to automate the Qlik Sense backup process), you can use the pgpass functionality in PostgreSQL. See the PostgreSQL documentation for more information.*

      iii. Make a backup of the dumpfile for the repository database.
4. Make a backup of log and application data in the following folders:

- *%ProgramData%\Qlik\Sense\Log*
- *%ProgramData%\Qlik\Sense\Apps*
- *%ProgramData%\Qlik\Sense\Repository\Content*
- *%ProgramData%\Qlik\Sense\Repository\Extensions*
- *%ProgramData%\Qlik\Sense\Repository\AppContent* (if available)

5. Make a backup of any locations where content that supports the Qlik Sense environment may be kept (for example, QVD files created by load scripts).
6. Start the Qlik Sense services. If the services are started manually, start them in the following order:
   a. Qlik Sense Repository Service (QRS)
      If the user running Qlik Sense services is not local administrator on the machine, you need to start Repository.exe from an elevated command prompt using the -bootstrap parameter.
      See: *Qlik Sense Repository Service (page 20)*
   b. Qlik Sense Proxy Service (QPS), Qlik Sense Engine Service (QES), Qlik Sense Scheduler Service (QSS), and Qlik Sense Printing Service (QPR) in no specific order

   The order is important because the QRS is dependent on the QRD and the rest of the services are dependent on the QRS.

Restoring a site manually

This section describes how to restore a Qlik Sense site in a default installation where a PostgreSQL database is used as the repository database.

> *The instructions in this section are only applicable to the central node of a Qlik Sense site.*
>
> *See: Backing up and restoring a site (page 59)*

The procedure described in this section can also be performed using the Repository Snapshot Manager (RSM).

See: *Using the Repository Snapshot Manager (page 66)*

Items to restore

The following items need to be considered when restoring a site:

- Qlik Sense software
- If you want to restore the site to a central node with a new hostname there are additional considerations.
  See: *Restoring a central node to a machine with a different hostname (page 66)*
- Repository database: The database contains all configuration data for the site.
- Certificates for the Qlik Sense services: The certificates are used to encrypt the traffic between the services and the users. Make sure to backup the certificates in order not to lose any encrypted data (for example, passwords for data connections).

- Log data
- Application data: The data models in the Qlik Sense apps.
- Any content that supports the apps (for example, QVD files)

Restoring a site with shared persistence

When performing the procedure below you must log on using an account that had the root admin role in the site when it was backed up. If you log on using a local admin account and the machine name is different, your permissions will not follow through.

Proceed as follows to restore a Qlik Sense site:

1. Install the Qlik Sense software on the machine targeted for the restore.
   See: Install and upgrade Qlik Sense

   > ⚠ *Make sure to deselect **Start the Qlik Sense services when the installation has completed** during the installation setup. If the services are started, new certificates and a new repository database are created and they must be removed before proceeding with the restore procedure.*

2. Restore the certificates used to secure the Qlik Sense services.
   See: *Restoring certificates (page 82)*

   > ⚠ *Do not start the Qlik Sense services at the end of the Restoring certificates (page 82) procedure.*

3. Start the Qlik Sense Repository Database (QRD).
4. Restore the repository database:
   a. Place the backed up repository database on the machine targeted for the restore.
   b. Open a Command Prompt with administrator privileges in Microsoft Windows.
   c. Run the following commands to restore the repository database (adjust the paths as needed):
      i. *cd "%ProgramFiles%\Qlik\Sense\Repository\PostgreSQL\<database version>\bin"*
      ii. *createdb -h localhost -p 4432 -U postgres -T template0 QSR*
      If the command fails because a database already exists, run the following command and then repeat the *createdb* command:
      dropdb -h localhost -p 4432 -U postgres QSR
      iii. *pg_restore.exe -h localhost -p 4432 -U postgres -d QSR "c:\QSR_backup.tar"*
5. Restore log and application data to the file share used for storage of log and application data.
6. Restore any supporting content to its original location as required.
7. Start the Qlik Sense services. If the services are started manually, start them in the following order:

a. Qlik Sense Repository Service (QRS)
If the user running Qlik Sense services is not local administrator on the machine, you need to start Repository.exe from an elevated command prompt using the -bootstrap parameter.
See: *Qlik Sense Repository Service (page 20)*

b. Qlik Sense Proxy Service (QPS), Qlik Sense Engine Service (QES), Qlik Sense Scheduler Service (QSS), and Qlik Sense Printing Service (QPR) in no specific order

The order is important because the QRS is dependent on the QRD and the rest of the services are dependent on the QRS.

Restoring a site with synchronized persistence

When performing the procedure below you must log on using an account that had the root admin role in the site when it was backed up. If you log on using a local admin account and the machine name is different, your permissions will not follow through.

Proceed as follows to restore a Qlik Sense site:

1. Install the Qlik Sense software on the machine targeted for the restore.

   See: Install and upgrade Qlik Sense

   > ⚠ *Make sure to deselect **Start the Qlik Sense services when the installation has completed** during the installation setup. If the services are started, new certificates and a new repository database are created and they must be removed before proceeding with the restore procedure.*

2. Restore the certificates used to secure the Qlik Sense services.

   See: *Restoring certificates (page 82)*

   > ⚠ *Do not start the Qlik Sense services at the end of the Restoring certificates (page 82) procedure.*

3. Start the Qlik Sense Repository Database (QRD).
4. Restore the repository database:
   a. Place the backed up repository database on the machine targeted for the restore.
   b. Open a Command Prompt with administrator privileges in Microsoft Windows.
   c. Run the following commands to restore the repository database (adjust the paths as needed):
      i. *cd "%ProgramFiles%\Qlik\Sense\Repository\PostgreSQL\<database version>\bin"*
      ii. *createdb -h localhost -p 4432 -U postgres -T template0 QSR*

         If the command fails because a database already exists, run the following command and then repeat the *createdb* command:
         dropdb -h localhost -p 4432 -U postgres QSR
      iii. *pg_restore.exe -h localhost -p 4432 -U postgres -d QSR "c:\QSR_backup.tar"*

5. Restore log and application data to these folders:
   - *%ProgramData%\Qlik\Sense\Log*
   - *%ProgramData%\Qlik\Sense\Apps*
   - *%ProgramData%\Qlik\Sense\Repository\Content*
   - *%ProgramData%\Qlik\Sense\Repository\Extensions*
   - *%ProgramData%\Qlik\Sense\Repository\AppContent* (if available)
6. Restore any supporting content to its original location as required.
7. Start the Qlik Sense services. If the services are started manually, start them in the following order:
   1. Qlik Sense Repository Service (QRS)
      If the user running Qlik Sense services is not local administrator on the machine, you need to start Repository.exe from an elevated command prompt using the -bootstrap parameter.
      See: *Qlik Sense Repository Service (page 20)*
   2. Qlik Sense Proxy Service (QPS), Qlik Sense Engine Service (QES), Qlik Sense Scheduler Service (QSS), and Qlik Sense Printing Service (QPR) in no specific order

   The order is important because the QRS is dependent on the QRD and the rest of the services are dependent on the QRS.
8. If you are restoring a multi-node site, you have two options:
   - **Resetting rim nodes**

     If a rim node does not contain any data that needs to be kept, remove and then re-add the rim node as if it was an entirely new node. This means that the content on the rim node will be the same as the content on the central node when it was restored.

     Do the following:

     1. In the Qlik Management Console (QMC) on the central node, delete the rim node.
     2. On the rim node, uninstall Qlik Sense with the **Remove Qlik Sense certificates and data folders** option selected.
     3. On the rim node, install Qlik Sense again.
     4. In the QMC on the central node, add the newly installed rim node to the site using the normal procedure.
   - **Keeping rim node data that was changed after backup**

     If a rim node contains data that was added after the backup, and is required to be kept, then the synchronization process needs to be notified to resend the missing data from the rim node to the restored central node. This also means that any data that was deleted on the rim node will be deleted on the restored site after synchronization.

     On the rim node, connect to the QRS API and call the *POST /qrs/sync/snapshot/restore* endpoint without any parameters or body. This endpoint instructs the Qlik Sense Repository Service (QRS) to push data back to the central node.

Restoring a central node to a machine with a different hostname

A restore can occur onto a machine with a different name than the one from which the data was backed up. However, if the machine is a central node in a multi-node site, you need to adapt the procedure.

- All rim nodes need to be reset, that is, you need to remove them and then add them again.
- Before starting the database, you need to update the hostname in the repository by executing the following command in PostgreSQL:
  ```
  UPDATE "LocalConfigs"
  SET "HostName" = '@newHostName'
  WHERE "HostName" = '@oldHostName';
  ```
  replacing @newHostName and @oldHostName with the actual new and old hostnames.
- Do not import certificates from the old system, as they will be invalid on the new node. You need to authorize certificates on the new node.
  You also need to update password information for all data connectors, as the passwords were encrypted using the certificate on the old system.

Known issues

If Qlik Sense stops responding or does not respond properly after the restore operation, try restarting the Qlik Sense Repository Service (QRS) and then wait for the rest of the services to start up.

See also:

🗋 *Backing up and restoring certificates (page 73)*

Using the Repository Snapshot Manager

The Repository Snapshot Manager (RSM) automates the manual backup and restore procedures described in the following sections:

🗋 *Backing up a site manually (page 59)*

🗋 *Restoring a site manually (page 62)*

> ⚠ *The Repository Snapshot Manager is only supported for use with synchronized persistence deployments. The Repository Snapshot Manager cannot be used to backup sites deployed with shared persistence. To backup sites deployed with shared persistence, see Backing up a site manually.*

Requirements

System downtime

The backup and restore operations require system downtime. The amount of time needed to back up or restore a site depends on the number of apps, their size, and the number of other supporting content files, such as QVD files and scripts.

Location

The RSM is available in *%ProgramFiles%\Qlik\Sense\Repository\Util\RepositorySnapshotManager* after installation of the Qlik Sense software.

> ⚠ *Do not run the RSM from the installation folder. Instead, copy the RSM to another folder and run it from there as restore operations delete the base installation folder %ProgramFiles%\Qlik\Sense recursively.*

User

The user who runs the RSM must be the same as the user who runs the Qlik Sense services.

Arguments

This section describes the arguments to pass to the RSM when backing up or restoring a Qlik Sense site.

Backup

The syntax of the RSM when backing up a Qlik Sense site is as follows:

RepositorySnapshotManager.exe -<Operation> -path=<Path> [-ver=<QlikSenseVersion>] [-supportingContentLocation=<PathToSupportingContent>] [-certificatePassword=<CertificatePassword>] [-databasePassword=<DatabasePassword>] [-f] [-h]

Argument	**Mandatory**	**Description**
-<Operation>	Yes	Type of operation. For backup, the argument is -backup.
-path	Yes	Path to the folder where to store the backup.
-ver	Optional, only needed if the automatic version check fails	Qlik Sense product version running at the time of the backup (for example, 2.2.0).
-supportingContentLocation	Optional, only needed if there is supporting content to include in the backup	Path to the folder in which supporting content for your Qlik Sense site is stored.
-certificatePassword	Optional, only needed if you want to password-protect the certificates in the backup	Password used to protect the certificates in the backup.
-databasePassword	Optional, only needed if a password was assigned to the repository database during the installation of the Qlik Sense software	Password assigned to the repository database during the installation of the Qlik Sense software. See: *Known issues (page 72)*

Argument	Mandatory	Description
-f	Optional, only needed if you want to run the backup operation silently	Silent backup. We recommend that you test your backup operation before using this argument.
-h (or -help)	Not applicable	Display information on the RSM. *This argument is not used to run an operation.*

Restore

The syntax of the RSM when restoring a Qlik Sense site is as follows:

RepositorySnapshotManager.exe -<Operation> -path=<Path> -installer=<PathToInstaller> [-supportingContentLocation=<PathToSupportingContent>] [-certificatePassword=<CertificatePassword>] [-databasePassword=<DatabasePassword>] [-u=<UsernameWithDomain>] [-p=<UserPassword>] [-installdir=<PathToAlternativeInstallDirectory>] [-h]

Argument	Mandatory	Description
-<Operation>	Yes	Type of operation. For restore, the argument is -restore.
-path	Yes	Path to the backup folder from which to restore the Qlik Sense site.
-installer	Yes	Path to the folder in which the Qlik Sense installer is located.
-supportingContentLocation	Optional, only needed if supporting content is to be restored to another location than from which it was backed up	Path to the restore location for the supporting content. If this argument is left out, the supporting content is restored to the location from which it was backed up.
-certificatePassword	Optional, only needed if the certificates were password-protected during the backup	Password that was used to protect the certificates in the backup.

Argument	Mandatory	Description
-databasePassword	Optional, only needed if you want to password-protect the repository database	Password used to protect the repository database.
-u	Optional, only needed if another user account than the local service account is to be used to restore and start the Qlik Sense software	User on the host with administrative privileges for the domain (for example, “COMPANY_DOMAIN\USER”). If no user credentials are provided, the Qlik Sense software will be restored and started using the local service account credentials.
-p	Optional, only needed if another user account than the local service account is to be used to restore and start the Qlik Sense software	Password for the user associated with the domain. If no user credentials are provided, the Qlik Sense software will be restored and started using the local service account credentials.
-installdir	Optional, only needed if you want to restore the Qlik Sense software to another folder than the default installation directory	Path to the folder in which to restore the Qlik Sense software. If this argument is left out, the default installation directory is used.
-h (or -help)	Not applicable	Display information on the RSM. *This argument is not used to run an operation.*

Procedures

Backup

Proceed as follows to back up a Qlik Sense site using the RSM:

1. Make sure that the Qlik Sense site is fully operational.
2. Schedule and prepare a system maintenance window with enough downtime.
3. Prepare a base backup folder (for example, *F:\Qlik\Backups*).
4. If the Qlik Sense repository database is password-protected, make sure that you have the password available.

   See: *Known issues (page 72)*

5. If you want to password-protect your certificates, make sure to pass the password as an argument to the RSM.
6. Gather all supporting content. All such content (for example, QVD files, scripts, and other files) must be stored in one location on the host where the RSM will be executed. Create a folder (for example, *F:\Qlik\Additional\Files*) and place the supporting content in that folder.
7. Run the RSM and perform the backup operation.

   > *Do not run the RSM from the installation folder. Instead, copy the RSM to another folder and run it from there as restore operations delete the base installation folder %ProgramFiles%\Qlik\Sense recursively.*

8. Wait for the backup operation to finish and review the log files.

   See: *Log files (page 72)*
9. Manually verify your backup before starting automated backups using the RSM.

Example: Backing up a site with a database password and supporting content (additional files)

RepositorySnapshotManager.exe -backup -path=F:\Qlik\Backups -supportingContentLocation=F:\Qlik\Additional\QVDFiles -databasePassword=dfgg45Fr800

> *You may need to provide quotes ("") around certain arguments for the command shell to interpret the arguments correctly. For example, if you have "&|<" or similar characters in a password, the entire password must be surrounded by quotes. This means that a password like test#% must be passed to the RSM as "test#%". In addition, do not use a backslash at the end of paths as this may invalidate the final quote.*

Restore

Proceed as follows to restore a Qlik Sense site using the RSM:

1. Make sure that the Qlik Sense installer, *Qlik_Sense_setup.exe*, is present in a folder on the current host.
2. Make sure that the versions of the Qlik Sense software to restore and the Qlik Sense installer are the same.
3. Locate the backup set (that is, backup folder) to use (for example, *F:\Qlik\Backups\.2015-07-02-12.57.10-Z.backup*).
4. If you want to password-protect your repository database, make sure to pass the password as an argument to the RSM.
5. If you password-protected your certificates during the backup, make sure to pass the password as an argument to the RSM.
6. Run the RSM and perform the restore operation.

> *Do not run the RSM from the installation folder. Instead, copy the RSM to another folder and run it from there as restore operations delete the base installation folder %ProgramFiles%\Qlik\Sense recursively.*

7. Wait for the restore operation to finish and review the log files.

> *During a restore operation a command prompt may open with the results from the database restore operations. Errors indicating that the object to drop did not exist can safely be ignored.*

> *If the restore operation is performed on a multi-node site, the log file should contain the text "snapshot/restore has been invoked on <host> (ok)" for each rim node. Verify that any content created on those rim nodes after the backup set was created is synchronized to the central node.*

8. Test the Qlik Sense site (for example, open the hub, reload an app, open a sheet, etc).
9. If the system is malfunctioning after the restore operation, contact Qlik Support and provide all log files in the *logs* folder.

   See: *Log files (page 72)*

Example: Restoring a site with a database password and supporting content

```
RepositorySnapshotManager.exe -restore -path=F:\Qlik\Backups\.2015-07-02-11.33.27-Z.backup -
installer=F:\Qlik\InstallerExtracted -supportingContentLocation=F:\Qlik\Additional\QVDFiles -
databasePassword=dfgg45Fr800 -u="DOM\MyUserName" -p="mYUSeRpAss100"
```

Example: Restoring a site without a database password or supporting content

```
RepositorySnapshotManager.exe -restore -path=F:\Qlik\Backups\.2015-07-02-11.33.27-Z.backup -
installer=F:\Qlik\InstallerExtracted -u="DOM\MyUserName" -p="mYUSeRpAss100"
```

> *You may need to provide quotes ("") around certain arguments for the command shell to interpret the arguments correctly. For example, if you have "&|<" or similar characters in a password, the entire password must be surrounded by quotes. This means that a password like test#% must be passed to the RSM as "test#%". In addition, do not use a backslash at the end of paths as this may invalidate the final quote.*

Log files

The RSM logs the backup and restore operations in a sub-folder, *logs*, of the folder in which the RSM runs. If the RSM cannot create the sub-folder in the current working folder, the sub-folder is created in the Windows temp folder (typically *C:\Windows\Temp*) instead.

Make sure to always check the log folder after a backup or restore operation, whether the operation was successful or not.

> *If the restore operation is performed on a multi-node site, the log file should contain the text "snapshot/restore has been invoked on <host> (ok)" for each rim node. Verify that any content created on those rim nodes after the backup set was created is synchronized to the central node.*

Return values

The RSM returns one of the following exit codes when the operation is completed:

- 0 - Success
- -1 - An error or exception occurred during the operation.
- 1 - Invalid parameter specified.

Known issues

Backup process completes "successfully" with wrong database password

If the password provided with the -databasePassword argument does not match the password of the repository database, the resulting backup file will be empty even though the backup process appears to have finished successfully. You must therefore verify that the *QSR_backup.tar* file in the *Database* folder of your backup is not empty.

Backup process hangs during "Creating login credentials file for database..."

The repository database is password-protected, but the -databasePassword argument was not provided.

Cannot start service on computer

If "Cannot start service QlikSenseRepositoryDatabase on computer" is reported at the command prompt or in the log file, it is most likely due to problems with the user credentials (that is, the username and password). The following circumstances are known to cause such problems:

- The Qlik Sense software was installed with user credentials, but is restored without them.
- The Qlik Sense software was installed without user credentials, but is restored with them.
- The username or password contains special characters, such as "&%$", and was not quoted during the backup or recovery operation.
- The username or password is incorrect.
- The Qlik Sense installer returned "Unable to create sibling process".
  If this happens, contact Qlik Support.

Log files stored outside the default locations

The RSM does not support backup and restore of Qlik Sense log files that are stored in other locations than the default ones. Such log files must be manually backed up and restored.

See: *Storage (page 122)*

3.2 Backing up and restoring certificates

It is recommended that you back up the certificates on the central node in a Qlik Sense site so that they can be restored, if needed.

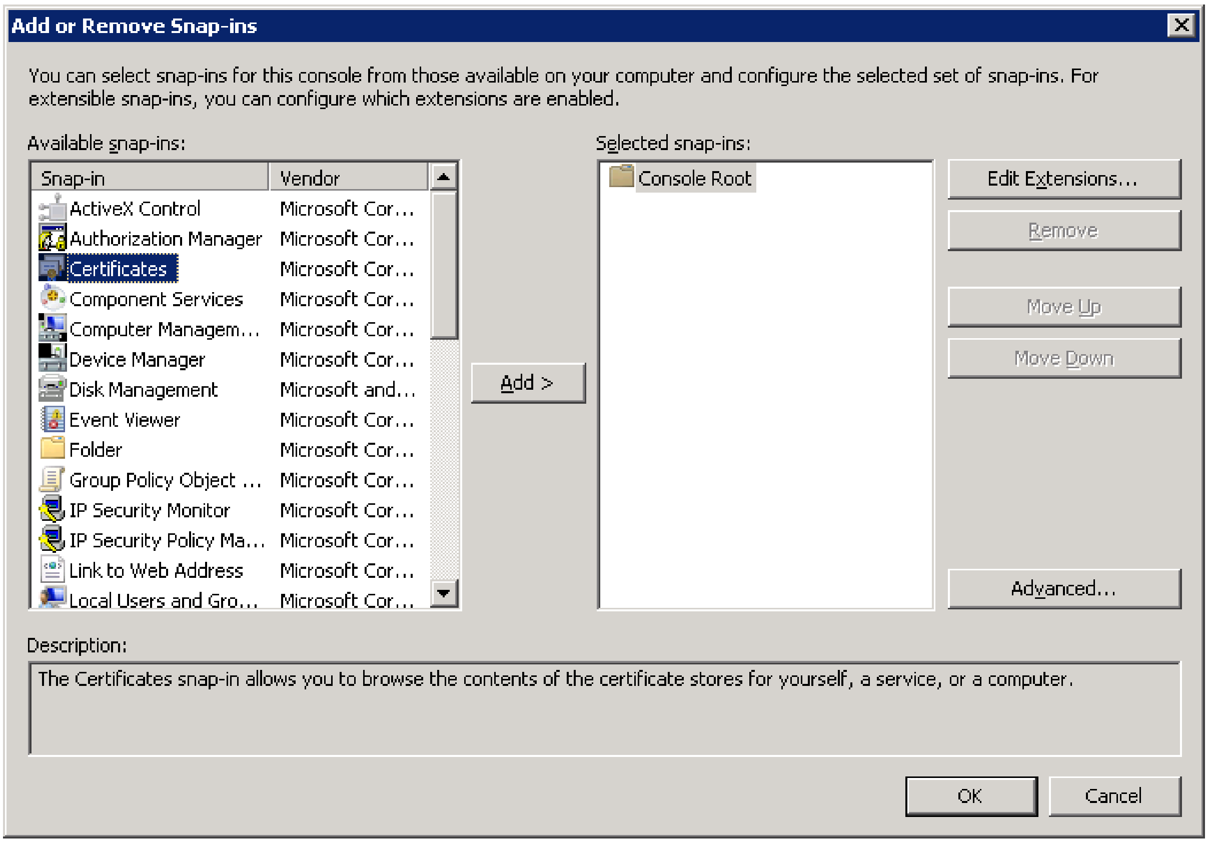
The backed up certificates can be used for different purposes:

- Restore the certificates on the **same** node as they were exported from.
- Move a node to **another** node in the site. This means that the repository database and its associated crypto key are reused on another node, but with new certificates for communication.

Backing up certificates

Proceed as follows to make a backup of the certificates on the central node in a Qlik Sense site:

1. Run *mmc* as the user that is used to run Qlik Sense services.
2. Select **File**>**Add/Remove Snap-in**.
3. Double-click **Certificates**.



Add or Remove Snap-ins

You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.

Available snap-ins:

Snap-in	Vendor
ActiveX Control	Microsoft Cor...
Authorization Manager	Microsoft Cor...
Certificates	Microsoft Cor...
Component Services	Microsoft Cor...
Computer Managem...	Microsoft Cor...
Device Manager	Microsoft Cor...
Disk Management	Microsoft and...
Event Viewer	Microsoft Cor...
Folder	Microsoft Cor...
Group Policy Object ...	Microsoft Cor...
IP Security Monitor	Microsoft Cor...
IP Security Policy Ma...	Microsoft Cor...
Link to Web Address	Microsoft Cor...
Local Users and Gro...	Microsoft Cor...

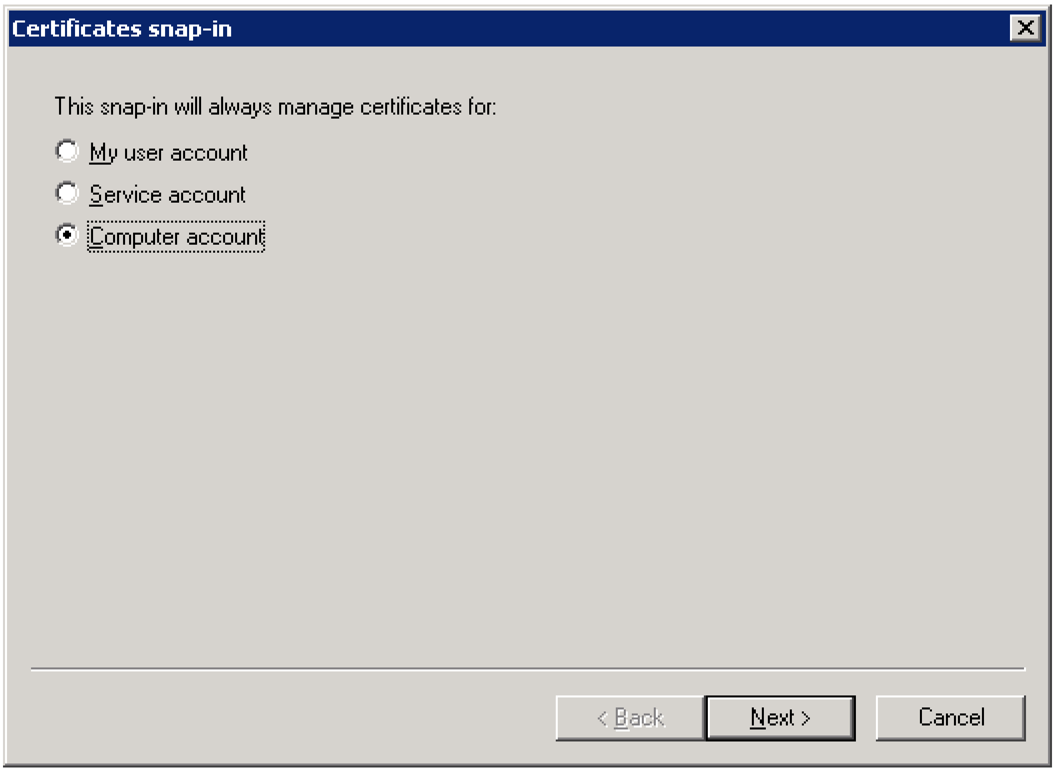
Selected snap-ins:

Console Root

Description:

The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.

4. Select **Computer account** and click **Next**.



Certificates snap-in

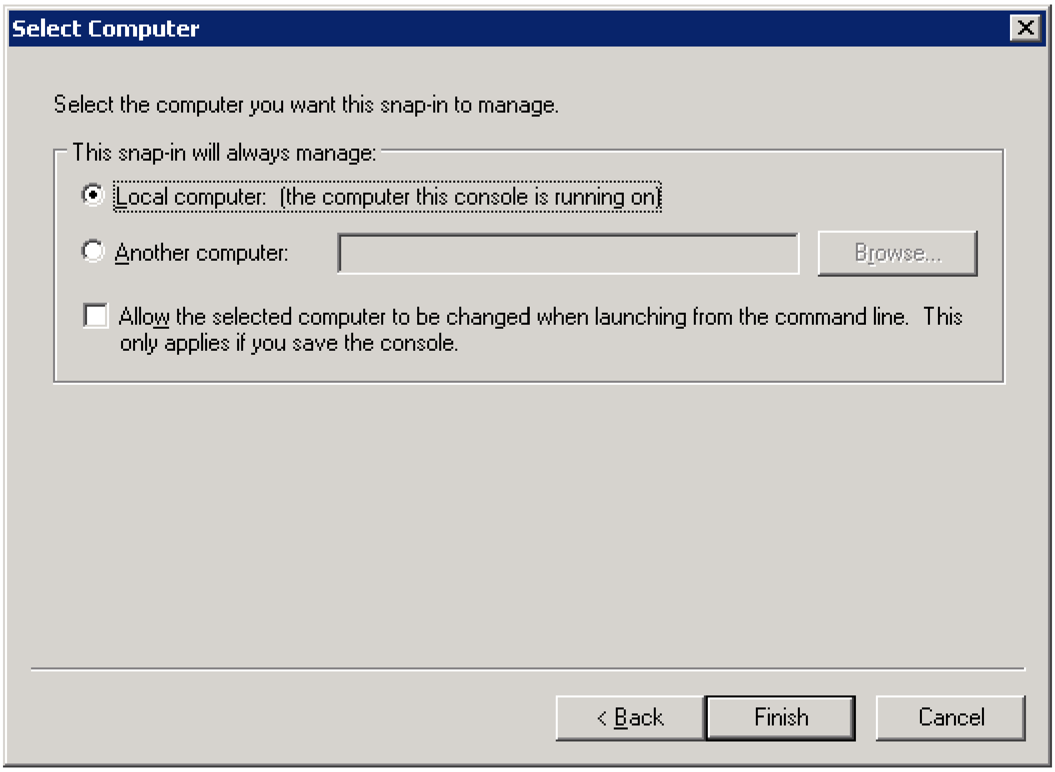
This snap-in will always manage certificates for:

○ My user account

○ Service account

● Computer account

5. Select **Local computer** and click **Finish**.



Select Computer

Select the computer you want this snap-in to manage.

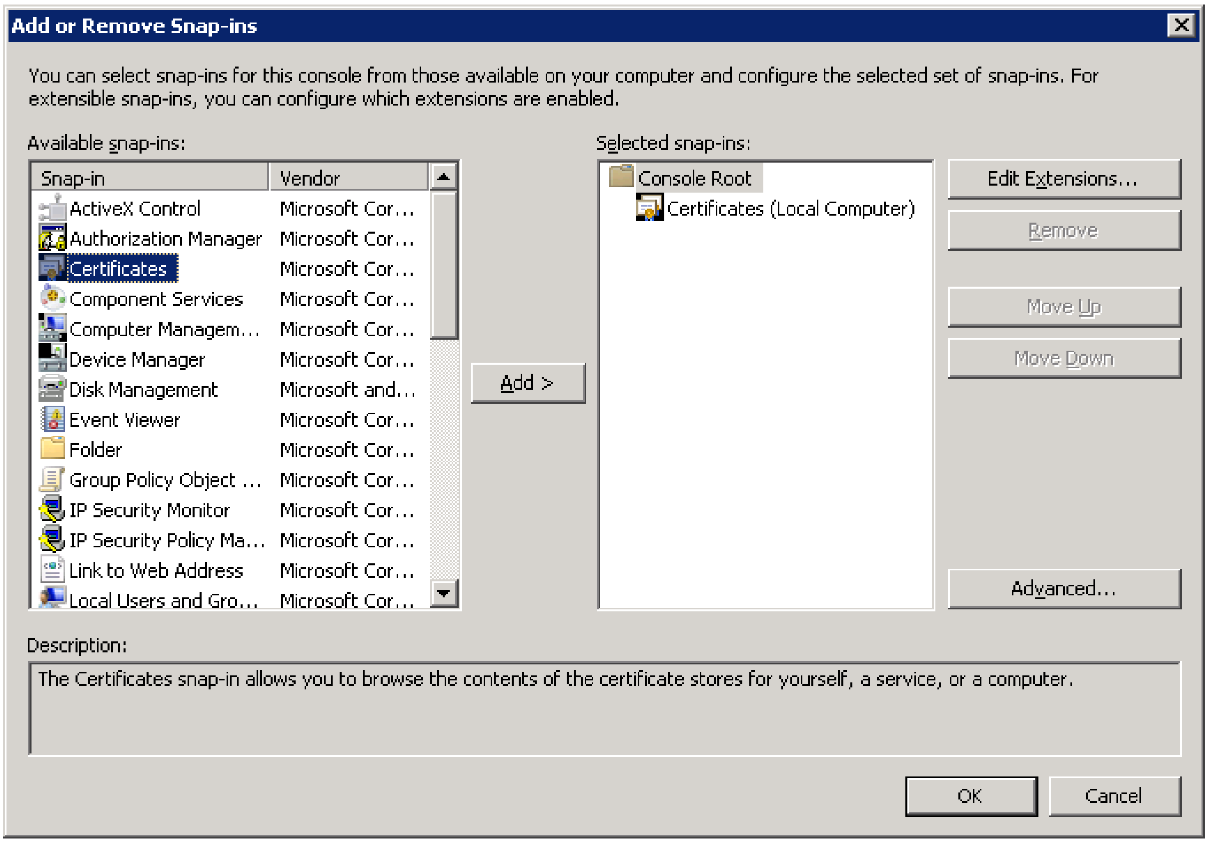
This snap-in will always manage:

● Local computer: (the computer this console is running on)

○ Another computer:

□ Allow the selected computer to be changed when launching from the command line. This only applies if you save the console.

6. Double-click **Certificates**.



Add or Remove Snap-ins

You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.

Available snap-ins:

Snap-in	Vendor
ActiveX Control	Microsoft Cor...
Authorization Manager	Microsoft Cor...
Certificates	Microsoft Cor...
Component Services	Microsoft Cor...
Computer Managem...	Microsoft Cor...
Device Manager	Microsoft Cor...
Disk Management	Microsoft and...
Event Viewer	Microsoft Cor...
Folder	Microsoft Cor...
Group Policy Object ...	Microsoft Cor...
IP Security Monitor	Microsoft Cor...
IP Security Policy Ma...	Microsoft Cor...
Link to Web Address	Microsoft Cor...
Local Users and Gro...	Microsoft Cor...

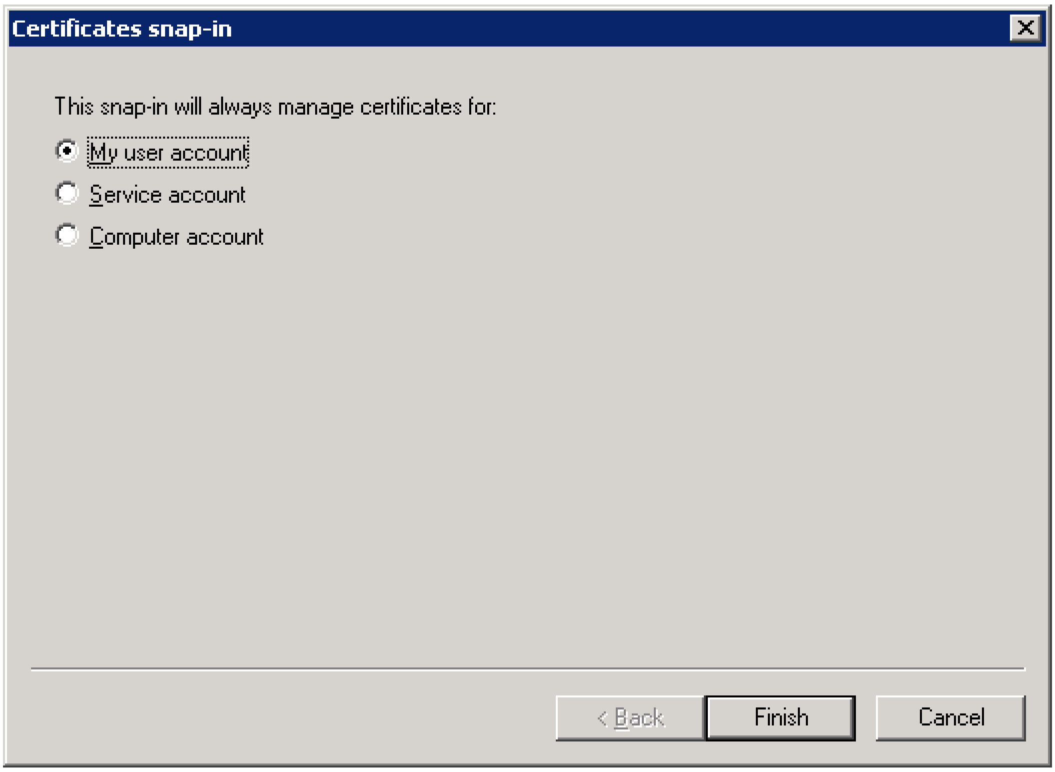
Selected snap-ins:

- Console Root
  - Certificates (Local Computer)

Description:

The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.

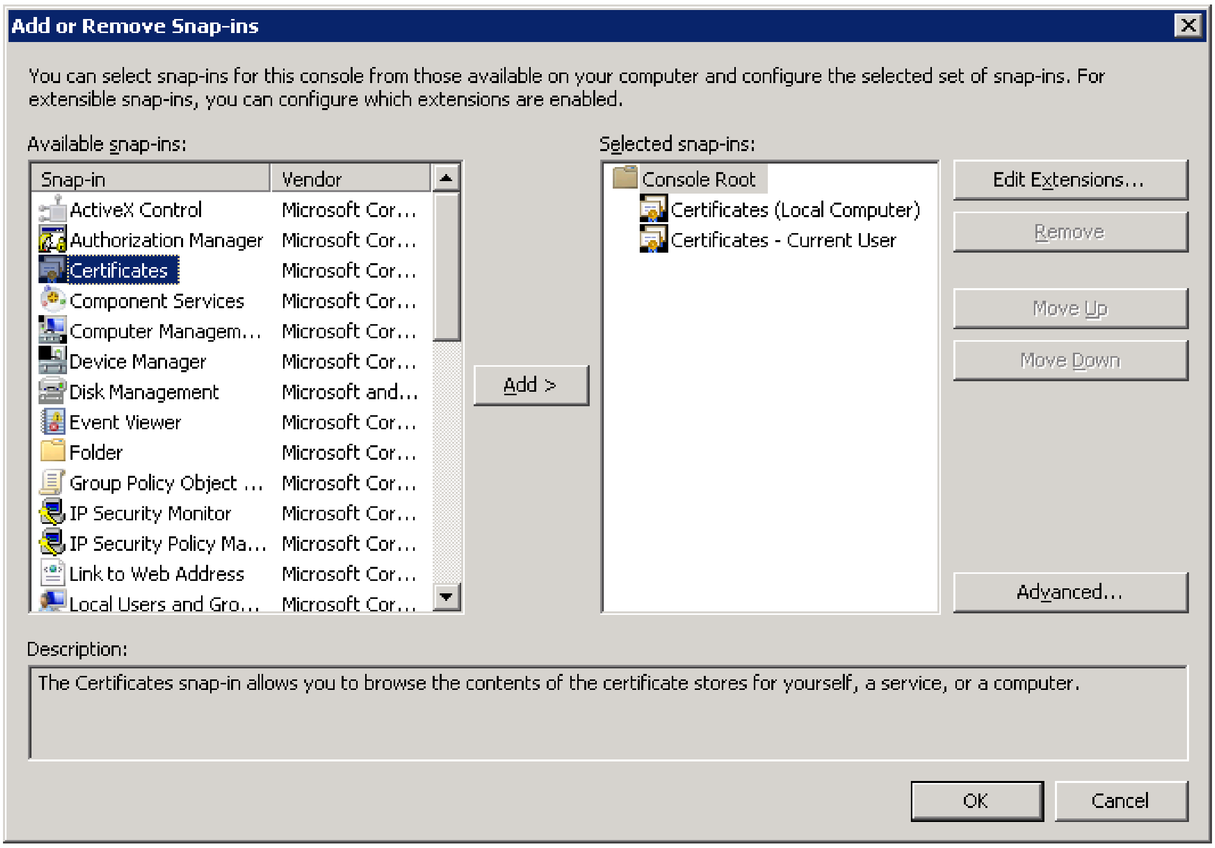
7. Select **My user account** and click **Finish**.



This snap-in will always manage certificates for:

● My user account
○ Service account
○ Computer account

8. Click **OK**.



You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.

Available snap-ins:

Snap-in	Vendor
ActiveX Control	Microsoft Cor...
Authorization Manager	Microsoft Cor...
Certificates	Microsoft Cor...
Component Services	Microsoft Cor...
Computer Managem...	Microsoft Cor...
Device Manager	Microsoft Cor...
Disk Management	Microsoft and...
Event Viewer	Microsoft Cor...
Folder	Microsoft Cor...
Group Policy Object ...	Microsoft Cor...
IP Security Monitor	Microsoft Cor...
IP Security Policy Ma...	Microsoft Cor...
Link to Web Address	Microsoft Cor...
Local Users and Gro...	Microsoft Cor...

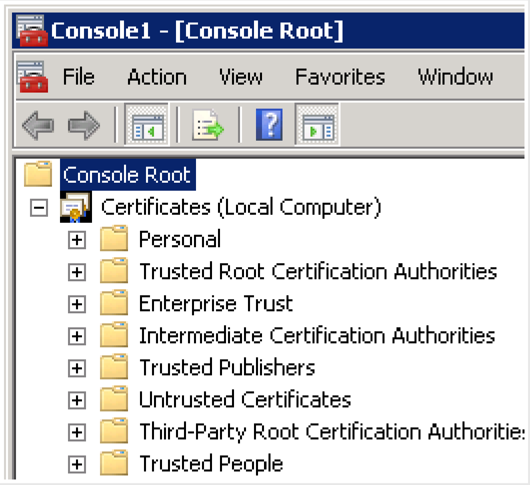
Selected snap-ins:

- Console Root
  - Certificates (Local Computer)
  - Certificates - Current User

Description:

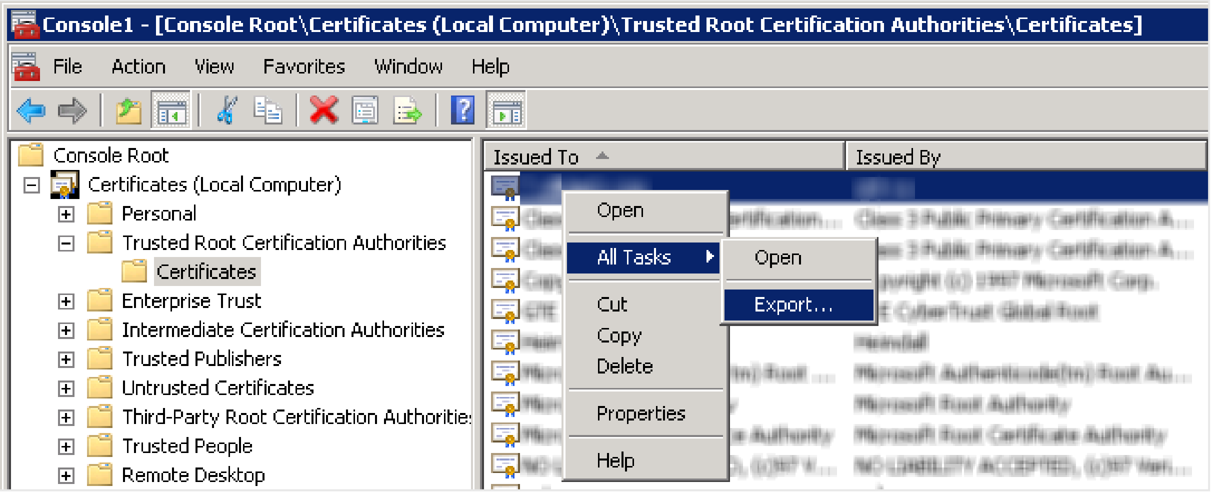
The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.

9. Expand **Certificates (Local Computer)** in the left panel.

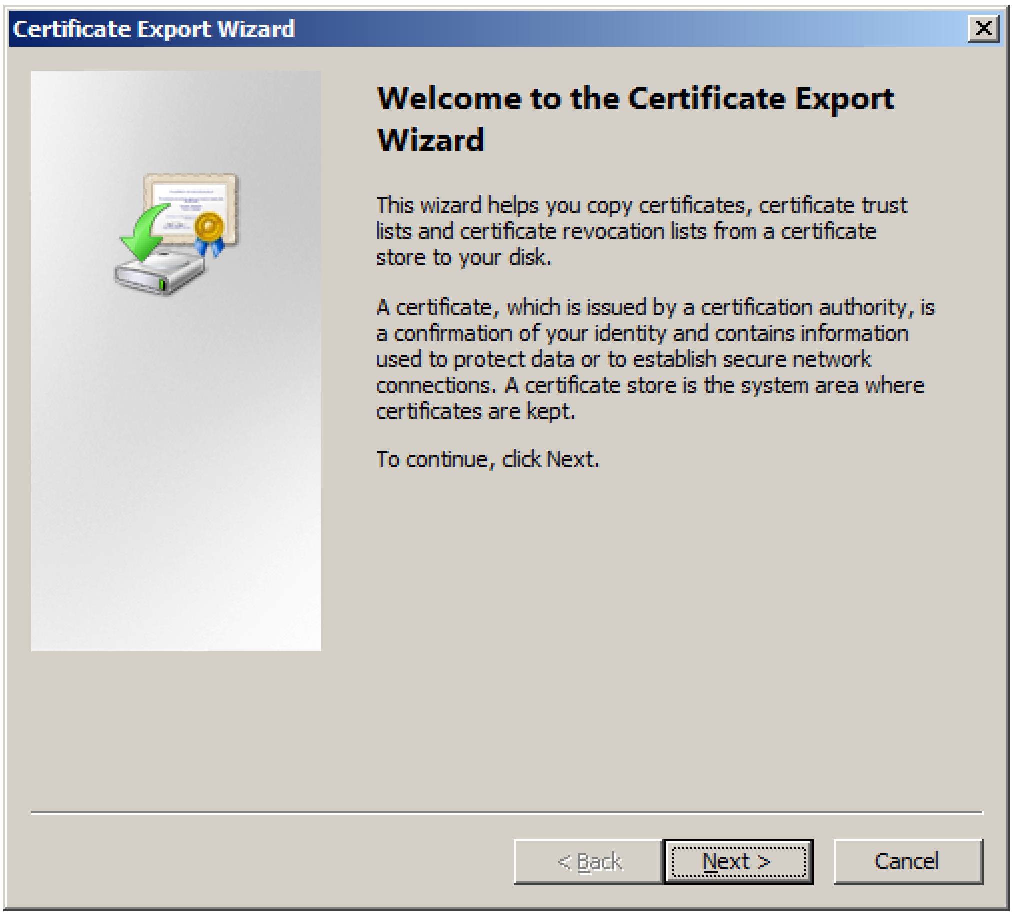

Console1 - [Console Root]
File
Action
View
Favorites
Window
Console Root
Certificates (Local Computer)
Personal
Trusted Root Certification Authorities
Enterprise Trust
Intermediate Certification Authorities
Trusted Publishers
Untrusted Certificates
Third-Party Root Certification Authoritie:
Trusted People

10. Expand the **Trusted Root Certification Authorities** folder and select the **Certificates** folder.

11. Right-click the certificate that is Certificate Authority (CA) for all nodes in the Qlik Sense site and select **All Tasks**>**Export**. The CA is named <*machine_that_issued_the_certificate*>*-CA* by default.


Console1 - [Console Root\Certificates (Local Computer)\Trusted Root Certification Authorities\Certificates]
File
Action
View
Favorites
Window
Help
Console Root
Certificates (Local Computer)
Personal
Trusted Root Certification Authorities
Certificates
Enterprise Trust
Intermediate Certification Authorities
Trusted Publishers
Untrusted Certificates
Third-Party Root Certification Authoritie:
Trusted People
Remote Desktop
Issued To
Issued By
Open
All Tasks
Cut
Copy
Delete
Properties
Help
Open
Export...

12. Click **Next**.



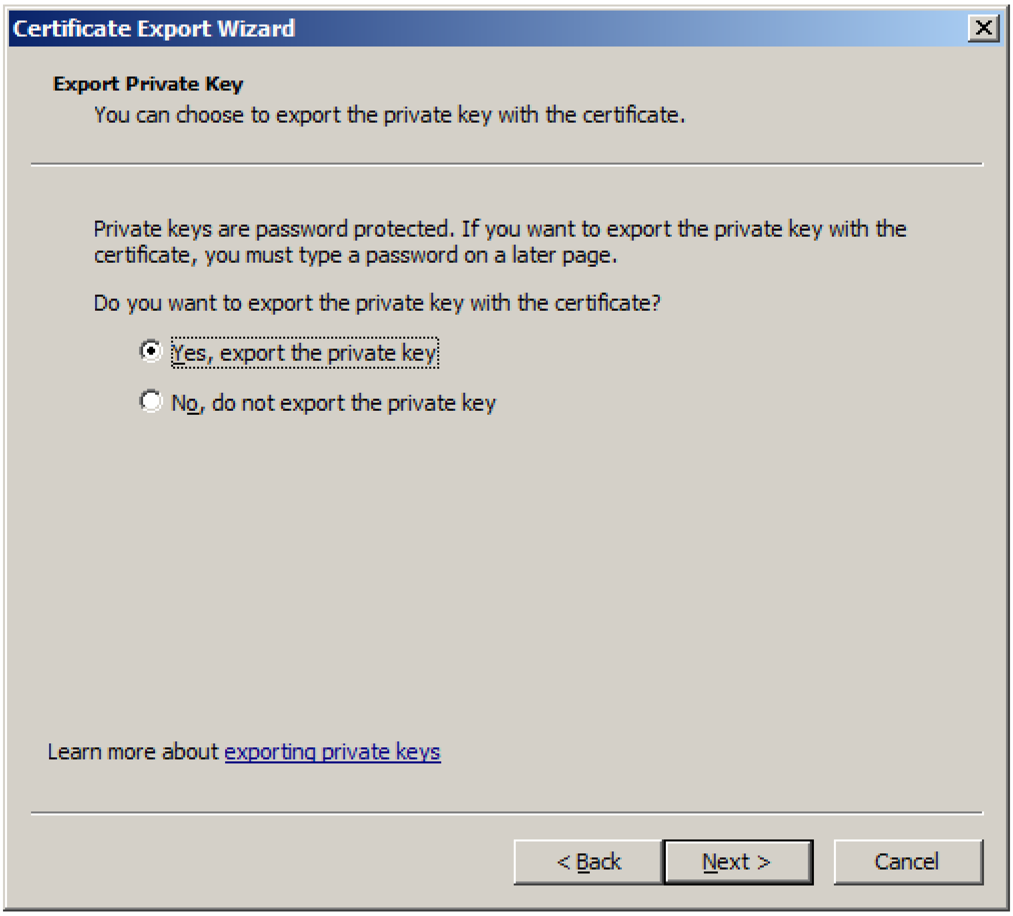
Welcome to the Certificate Export Wizard

This wizard helps you copy certificates, certificate trust lists and certificate revocation lists from a certificate store to your disk.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

To continue, click Next.

13. Select **Yes, export the private key** and click **Next**.



Export Private Key

You can choose to export the private key with the certificate.

Private keys are password protected. If you want to export the private key with the certificate, you must type a password on a later page.

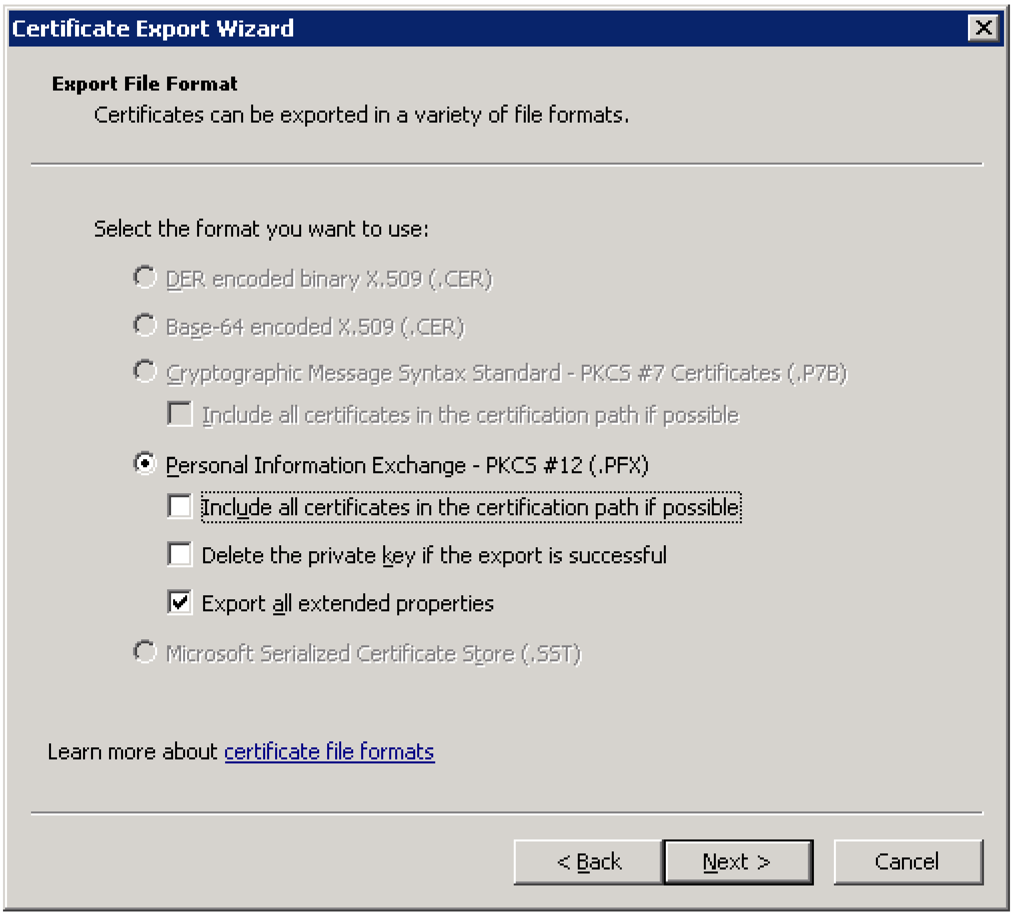
Do you want to export the private key with the certificate?

● Yes, export the private key

○ No, do not export the private key

Learn more about exporting private keys

14. Select **Personal Information Exchange**.
15. Tick the **Export all extended properties** box and then click **Next**.



Certificate Export Wizard

Export File Format

Certificates can be exported in a variety of file formats.

Select the format you want to use:

○ DER encoded binary X.509 (.CER)

○ Base-64 encoded X.509 (.CER)

○ Cryptographic Message Syntax Standard - PKCS #7 Certificates (.P7B)

□ Include all certificates in the certification path if possible

● Personal Information Exchange - PKCS #12 (.PFX)

□ Include all certificates in the certification path if possible

□ Delete the private key if the export is successful

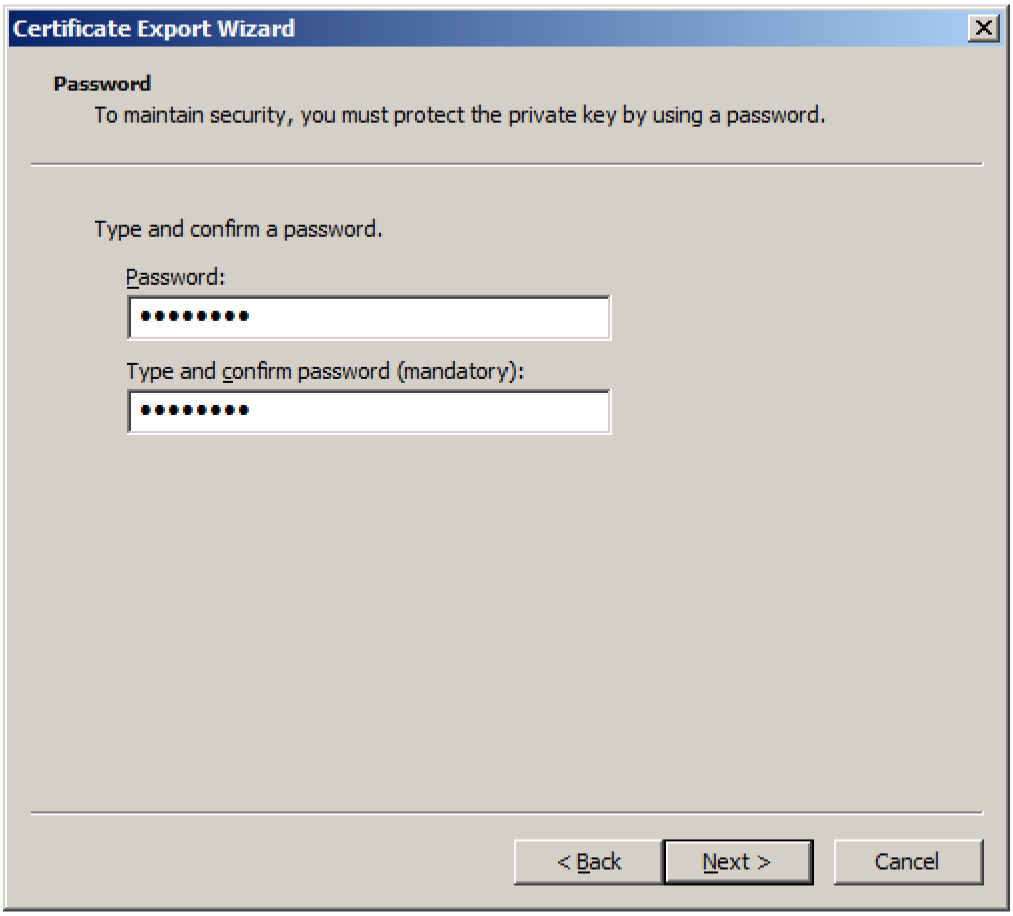
☑ Export all extended properties

○ Microsoft Serialized Certificate Store (.SST)

Learn more about certificate file formats

16. Enter and confirm a password. Then click **Next**.

    The password is needed when importing the certificate.



Certificate Export Wizard

Password

To maintain security, you must protect the private key by using a password.

Type and confirm a password.

Password:

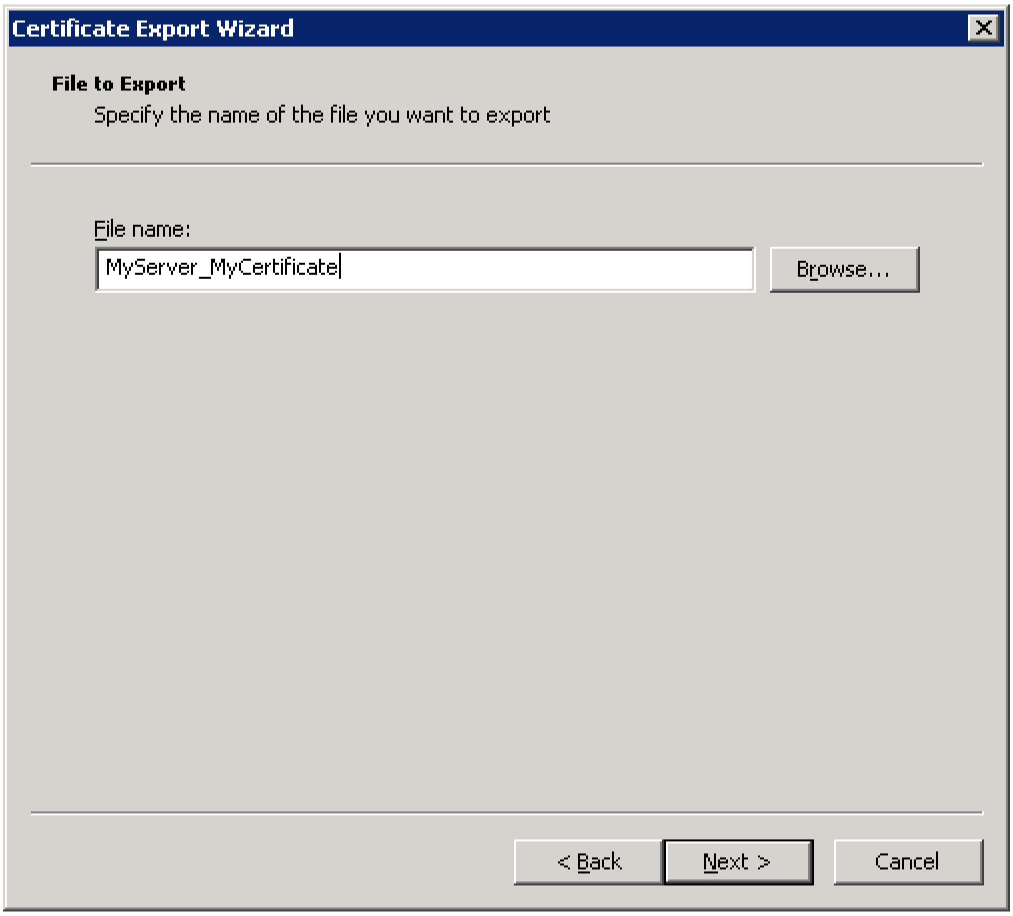
••••••••

Type and confirm password (mandatory):

••••••••

17. Enter a file name for the *.pfx* file and click **Next**.

    It is recommended to include the server name in the file name to avoid confusion with other certificate files.



Certificate Export Wizard

File to Export

Specify the name of the file you want to export

File name: MyServer_MyCertificate

18. Click **Finish**.

    The *.pfx* file that contains the CA for all nodes in the Qlik Sense site is stored in the selected location.

19. Starting at step 11, repeat the procedure and export the server certificate (that is, the SSL certificate), which is located under **Certificates (Local Computer)**>**Personal**>**Certificates**. The server certificate a) has the same name as the Domain Name System (DNS) name of the machine, and b) is signed by the CA for all nodes in the site.
20. Starting at step 11, repeat the procedure and export the client certificate (that is, the ID of the client), which is located under **Certificates - Current User**>**Personal**>**Certificates**. The client certificate is named *QlikClient* and is signed by the CA for all nodes in the site.
21. Close the MMC console.

    No changes have to be saved.

Restoring certificates

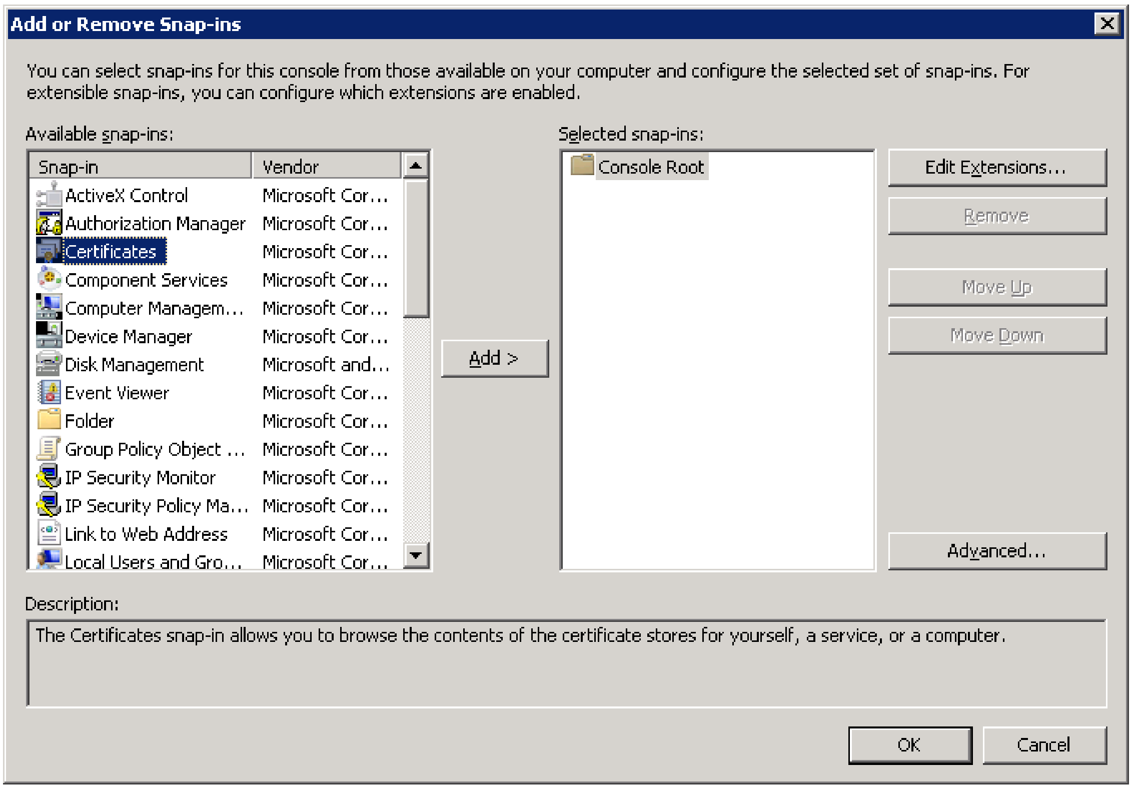
In case of a system crash, the certificates may have to be restored on the central node in the Qlik Sense site.

Proceed as follows to restore the certificates on the central node in a site:

1. Open the Task Manager in Microsoft Windows and stop all Qlik Sense services except the Qlik Sense Repository Database (QRD) service.

If you are restoring the certificates as part of the Restoring a site manually (page 62) procedure, skip this step.

2. Run *mmc* as the user that is used to run Qlik Sense services.
3. Select **File**>**Add/Remove Snap-in**.
4. Double-click **Certificates**.



Add or Remove Snap-ins

You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.

Available snap-ins:

Snap-in	Vendor
ActiveX Control	Microsoft Cor...
Authorization Manager	Microsoft Cor...
Certificates	Microsoft Cor...
Component Services	Microsoft Cor...
Computer Managem...	Microsoft Cor...
Device Manager	Microsoft Cor...
Disk Management	Microsoft and...
Event Viewer	Microsoft Cor...
Folder	Microsoft Cor...
Group Policy Object ...	Microsoft Cor...
IP Security Monitor	Microsoft Cor...
IP Security Policy Ma...	Microsoft Cor...
Link to Web Address	Microsoft Cor...
Local Users and Gro...	Microsoft Cor...

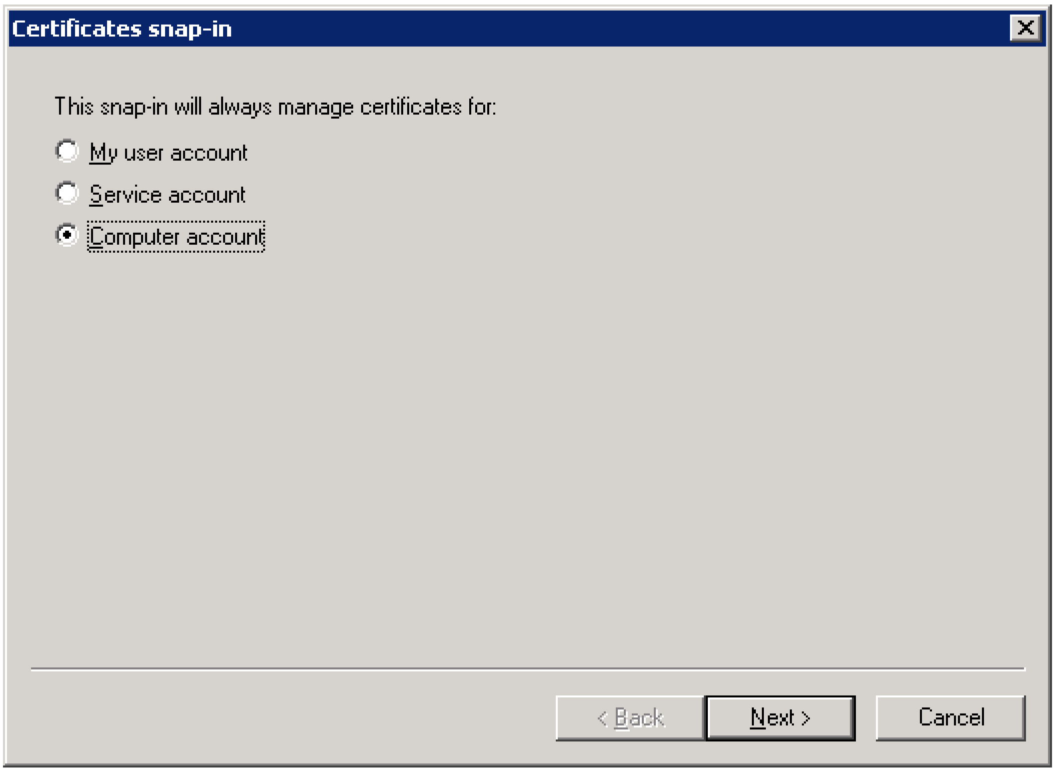
Selected snap-ins:

Console Root

Description:

The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.

5. Select **Computer account** and click **Next**.



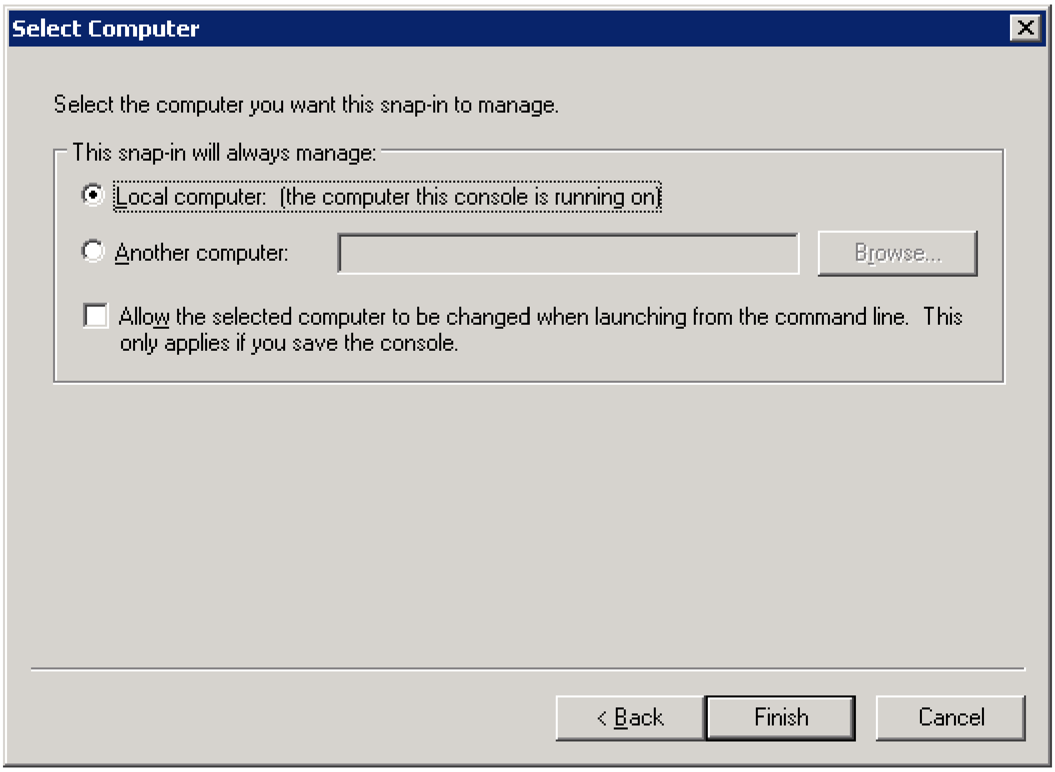
This snap-in will always manage certificates for:

○ My user account

○ Service account

● Computer account

6. Select **Local computer** and click **Finish**.



Select the computer you want this snap-in to manage.

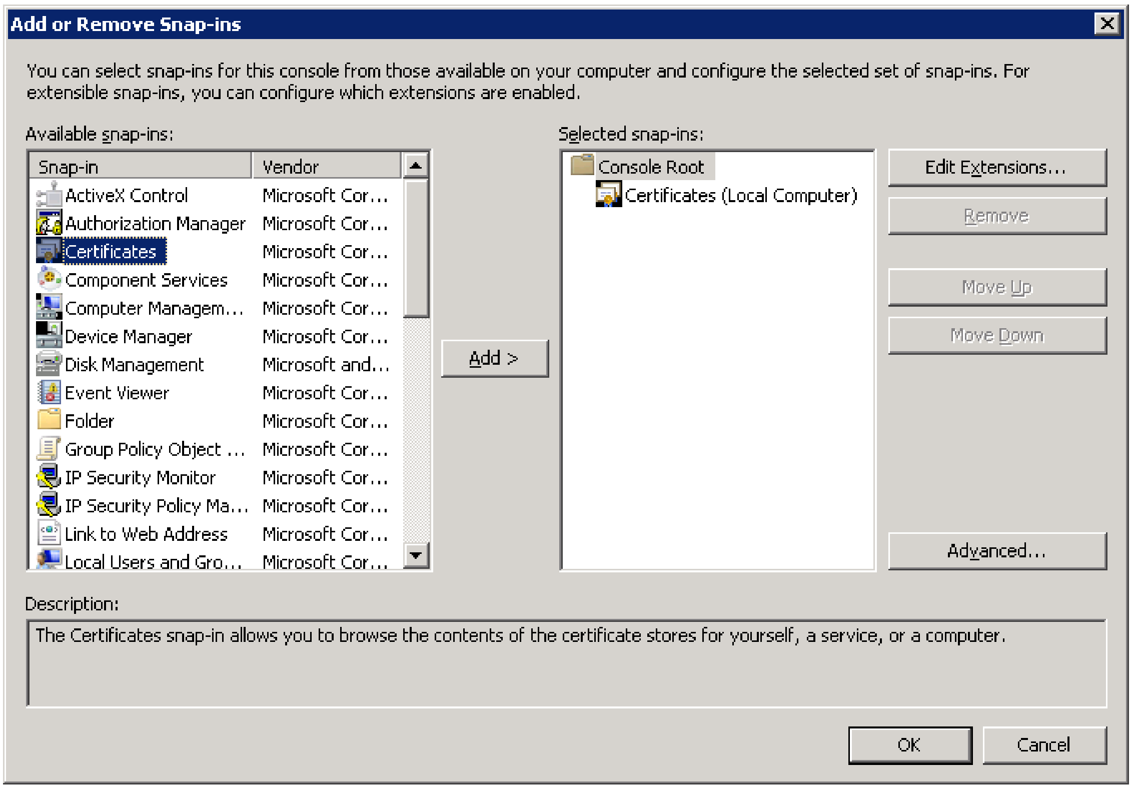
This snap-in will always manage:

● Local computer: (the computer this console is running on)

○ Another computer:

□ Allow the selected computer to be changed when launching from the command line. This only applies if you save the console.

7. Double-click **Certificates**.



Add or Remove Snap-ins

You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.

Available snap-ins:

Snap-in	Vendor
ActiveX Control	Microsoft Cor...
Authorization Manager	Microsoft Cor...
Certificates	Microsoft Cor...
Component Services	Microsoft Cor...
Computer Managem...	Microsoft Cor...
Device Manager	Microsoft Cor...
Disk Management	Microsoft and...
Event Viewer	Microsoft Cor...
Folder	Microsoft Cor...
Group Policy Object ...	Microsoft Cor...
IP Security Monitor	Microsoft Cor...
IP Security Policy Ma...	Microsoft Cor...
Link to Web Address	Microsoft Cor...
Local Users and Gro...	Microsoft Cor...

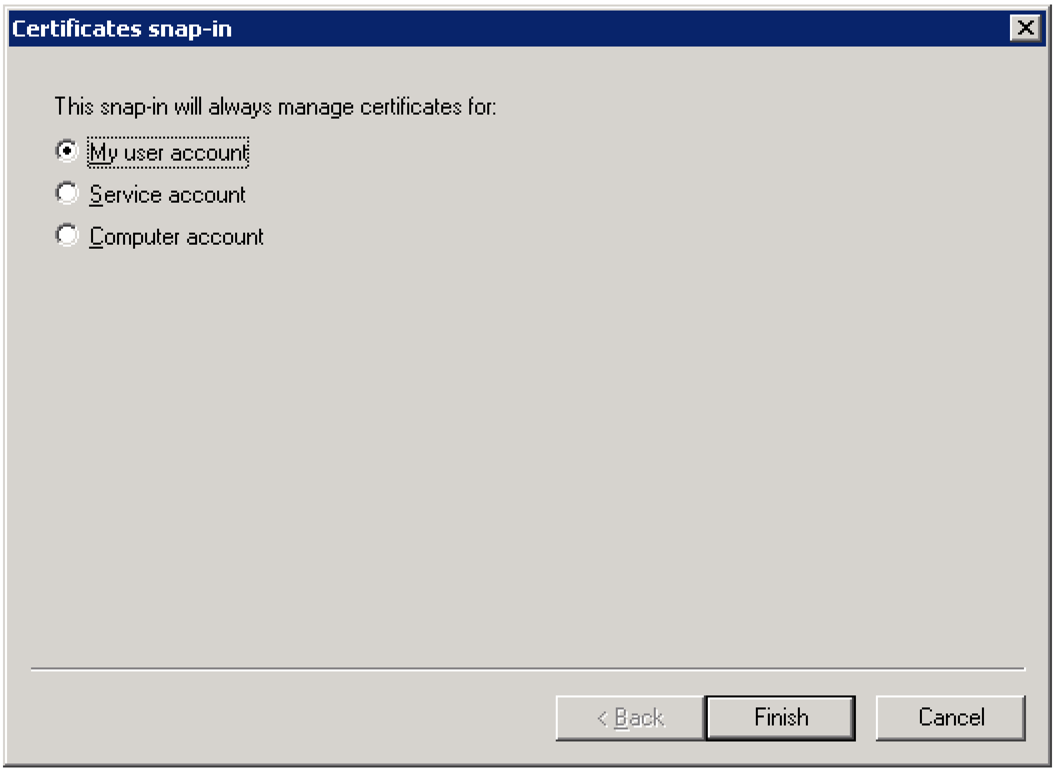
Selected snap-ins:

Console Root
Certificates (Local Computer)

Description:

The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.

8. Select **My user account** and click **Finish**.



Certificates snap-in

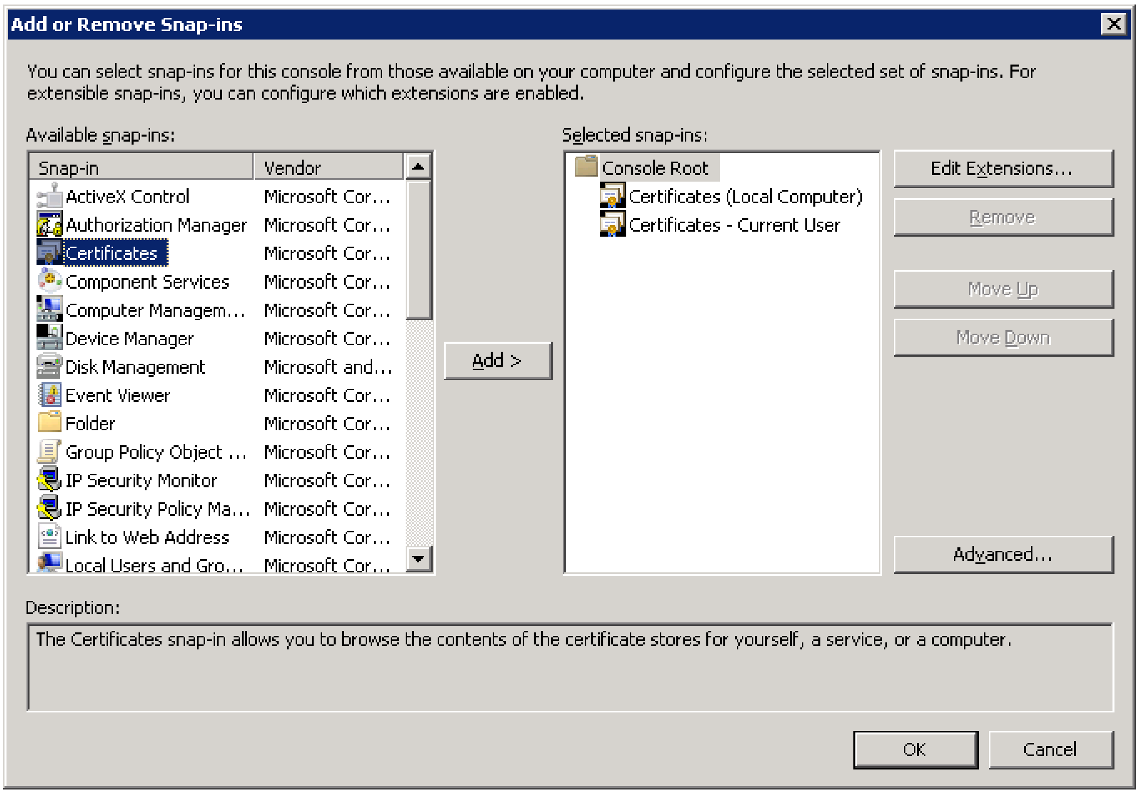
This snap-in will always manage certificates for:

● My user account

○ Service account

○ Computer account

9. Click **OK**.



Add or Remove Snap-ins

You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.

Available snap-ins:

Snap-in	Vendor
ActiveX Control	Microsoft Cor...
Authorization Manager	Microsoft Cor...
Certificates	Microsoft Cor...
Component Services	Microsoft Cor...
Computer Managem...	Microsoft Cor...
Device Manager	Microsoft Cor...
Disk Management	Microsoft and...
Event Viewer	Microsoft Cor...
Folder	Microsoft Cor...
Group Policy Object ...	Microsoft Cor...
IP Security Monitor	Microsoft Cor...
IP Security Policy Ma...	Microsoft Cor...
Link to Web Address	Microsoft Cor...
Local Users and Gro...	Microsoft Cor...

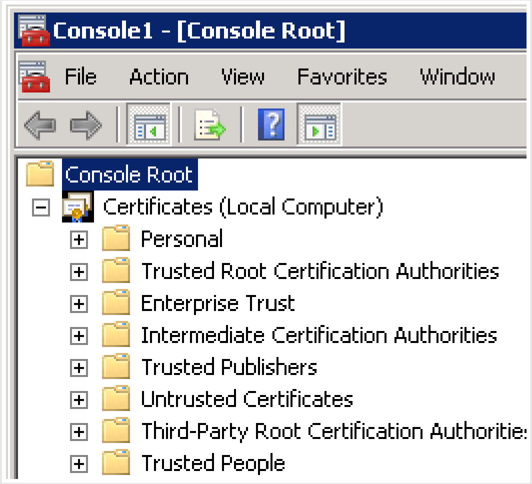
Selected snap-ins:

- Console Root
  - Certificates (Local Computer)
  - Certificates - Current User

Description:

The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.

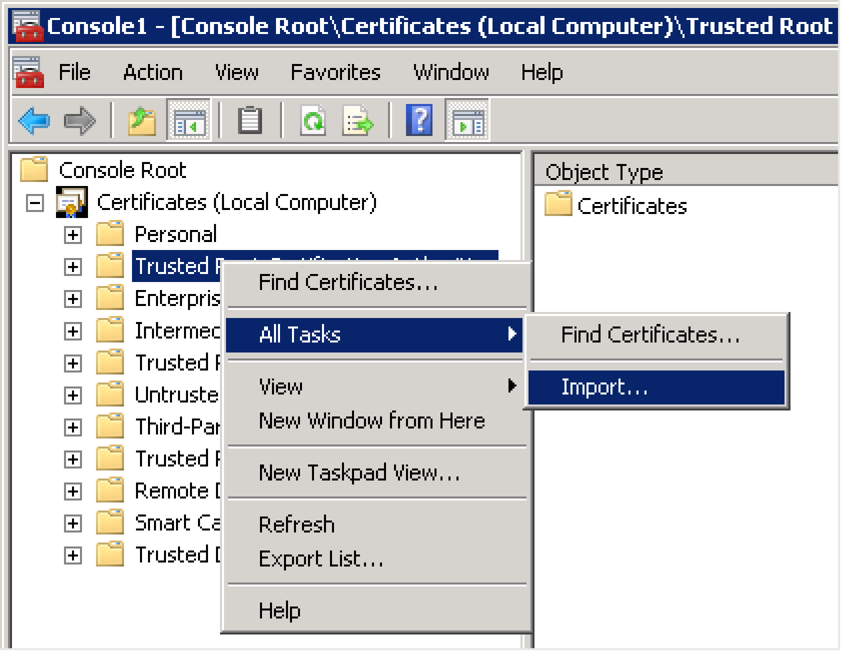
10. Expand **Certificates (Local Computer)** in the left panel.



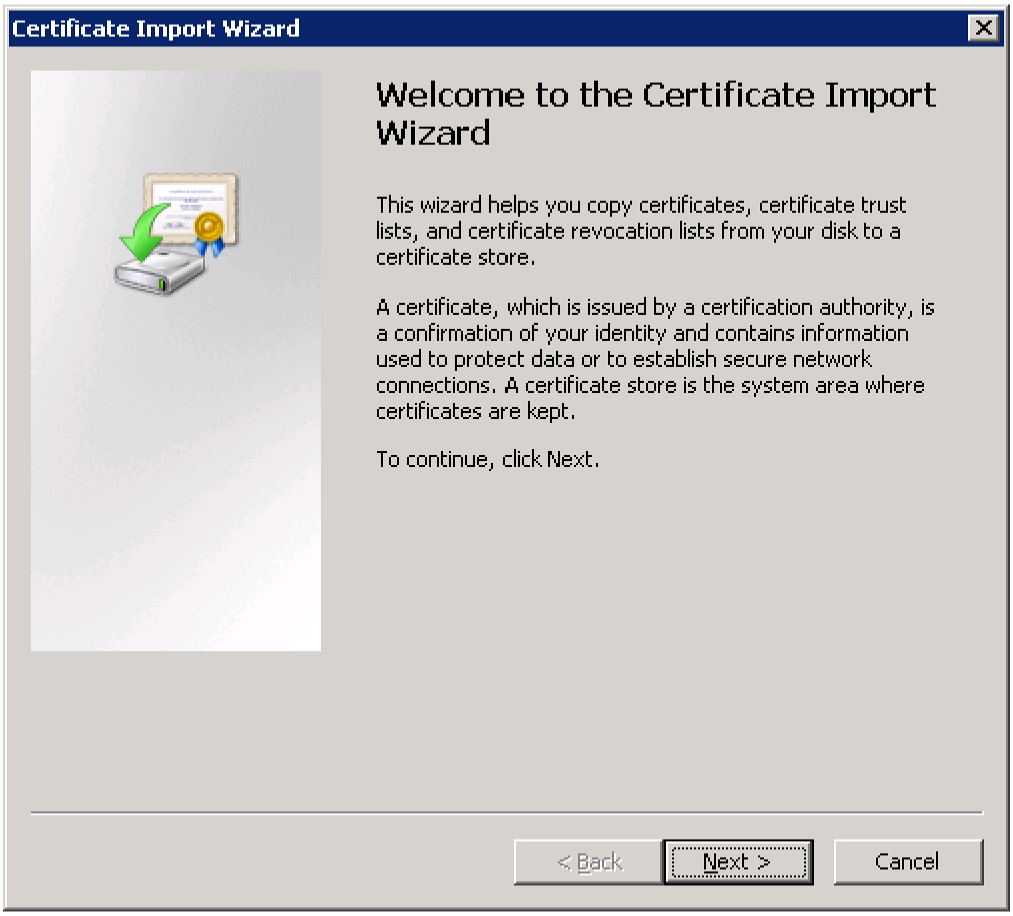
Console1 - [Console Root]

- Console Root
  - Certificates (Local Computer)
    - Personal
    - Trusted Root Certification Authorities
    - Enterprise Trust
    - Intermediate Certification Authorities
    - Trusted Publishers
    - Untrusted Certificates
    - Third-Party Root Certification Authoritie
    - Trusted People

11. Right-click the **Trusted Root Certification Authorities** folder and select **All Tasks**>**Import**.



12. Click **Next**.



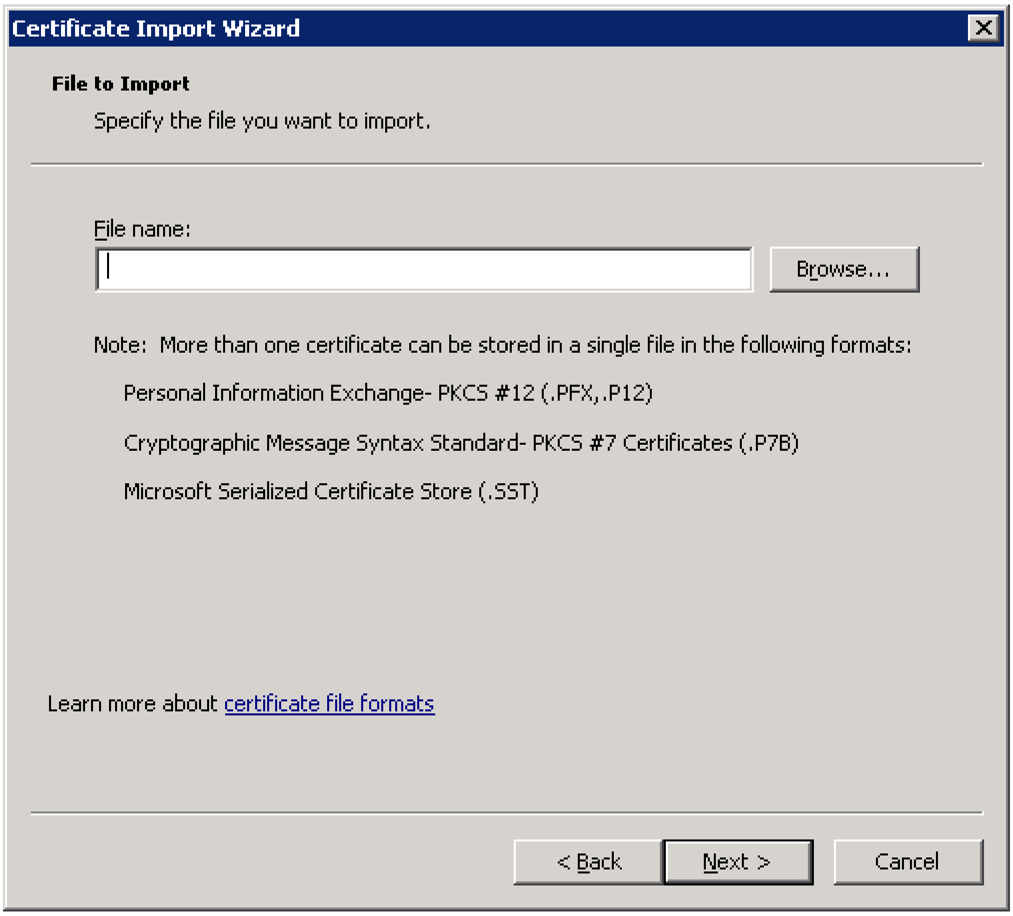
Welcome to the Certificate Import Wizard

This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

To continue, click Next.

13. Browse to the file that contains the backed up Certificate Authority (CA) for all nodes in the site and then click **Next**. The CA is named <*machine_that_issued_the_certificate*>*-CA* by default.



Certificate Import Wizard

File to Import

Specify the file you want to import.

File name:

Note: More than one certificate can be stored in a single file in the following formats:

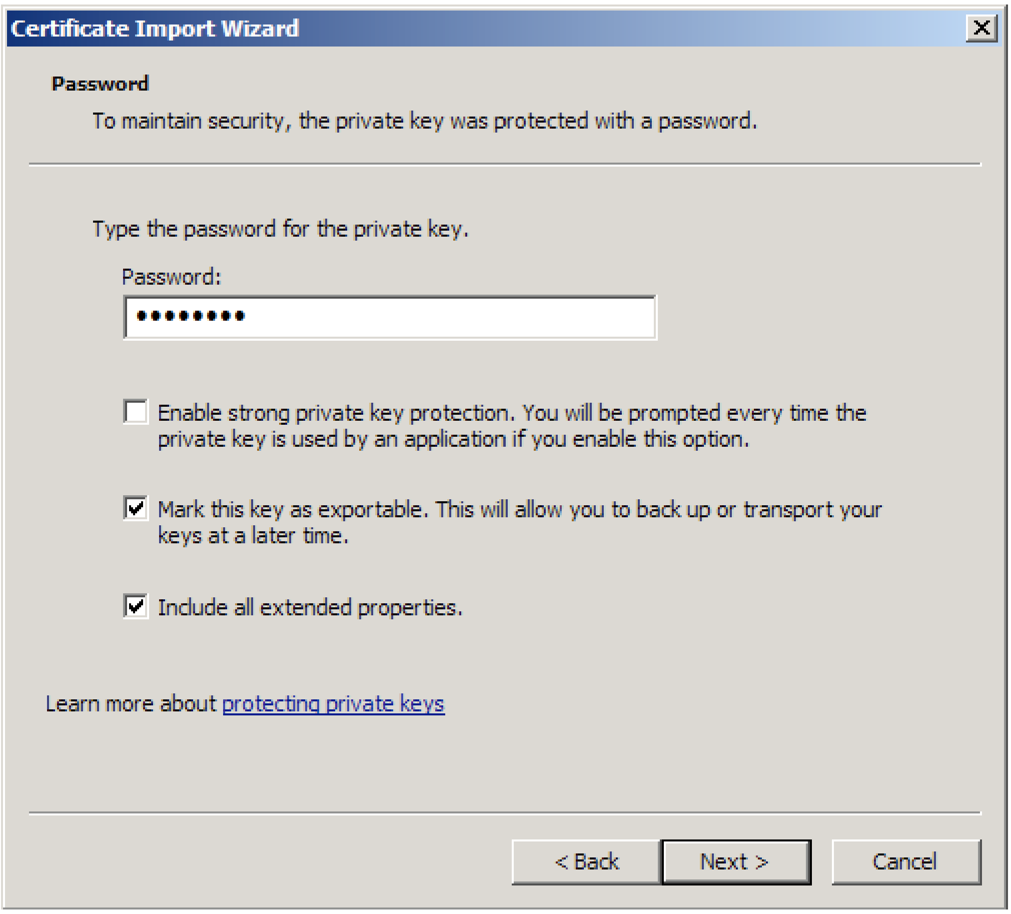
Personal Information Exchange- PKCS #12 (.PFX,.P12)

Cryptographic Message Syntax Standard- PKCS #7 Certificates (.P7B)

Microsoft Serialized Certificate Store (.SST)

Learn more about certificate file formats

14. Enter the password for the *.pfx* file (that is, the password that was given when the file was exported).
15. Select **Mark this key as exportable** and **Include all extended properties**. Then click **Next**.



Password

To maintain security, the private key was protected with a password.

Type the password for the private key.

Password:

●●●●●●●●●

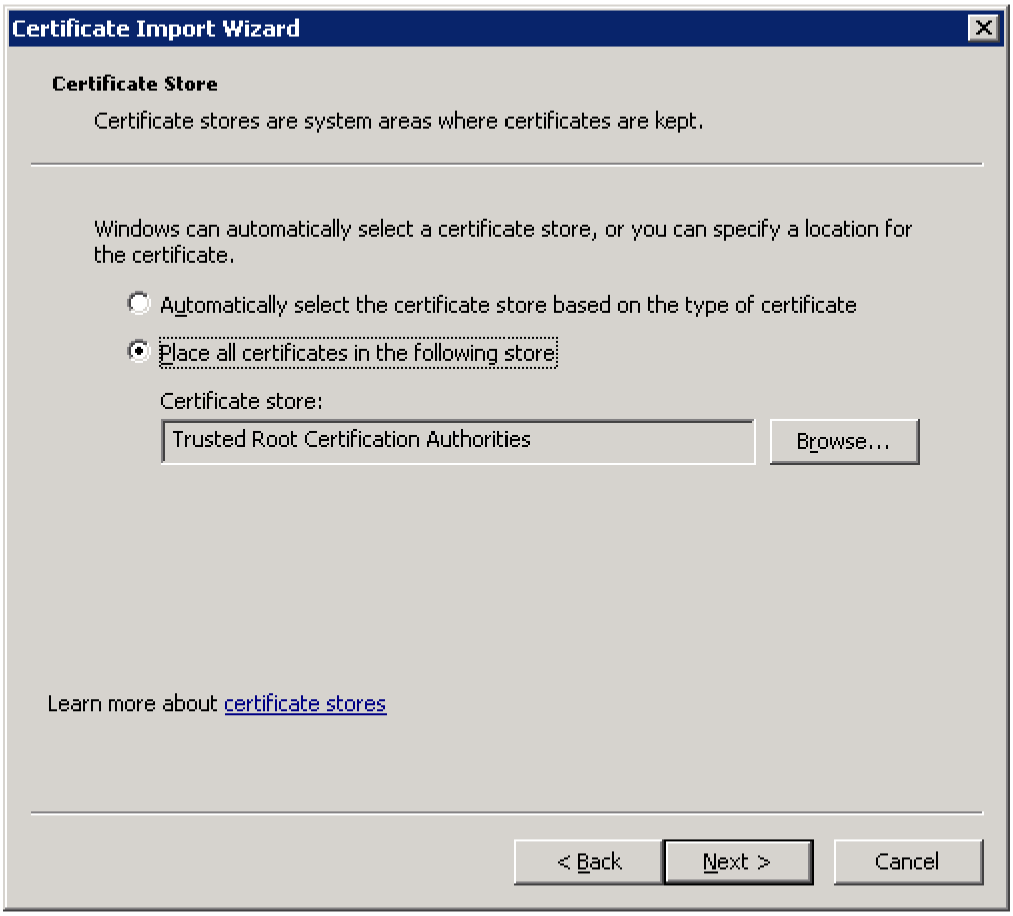
□ Enable strong private key protection. You will be prompted every time the private key is used by an application if you enable this option.

☑ Mark this key as exportable. This will allow you to back up or transport your keys at a later time.

☑ Include all extended properties.

Learn more about protecting private keys

16. Select **Place all certificates in the following store** and click **Next**.



Certificate Import Wizard

Certificate Store

Certificate stores are system areas where certificates are kept.

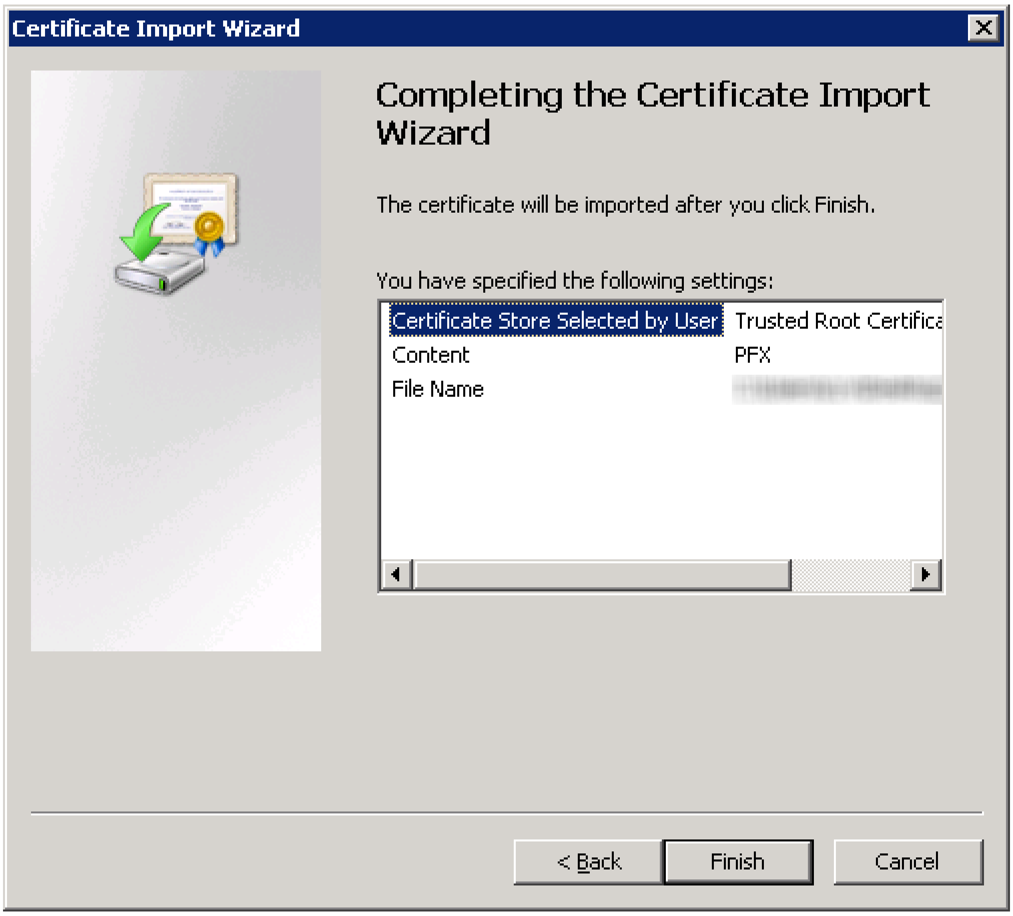
Windows can automatically select a certificate store, or you can specify a location for the certificate.

○ Automatically select the certificate store based on the type of certificate

● Place all certificates in the following store

Certificate store: Trusted Root Certification Authorities

17. Click **Finish**.



Certificate Import Wizard

Completing the Certificate Import Wizard

The certificate will be imported after you click Finish.

You have specified the following settings:

Certificate Store Selected by User	Trusted Root Certifica
Content	PFX
File Name	

18. Click the **Refresh** button (🔄) and check that the imported CA is available in the **Trusted Root Certification Authorities** folder.
19. Starting at step 11, repeat the procedure and import the server certificate to **Certificates (Local Computer)**>**Personal**>**Certificates**. The server certificate a) has the same name as the Domain Name System (DNS) name of the machine, and b) is signed by the CA for all nodes in the site.
20. Starting at step 11, repeat the procedure and import the client certificate (that is, the ID of the client) to **Certificates - Current User**>**Personal**>**Certificates**. The client certificate is named *QlikClient* and is signed by the CA for all nodes in the site.
21. Close the MMC console.

    No changes have to be saved.

22. Start the Qlik Sense services. If the services are started manually, start them in the following order:

    ⚠ *If you are restoring the certificates as part of the Restoring a site manually (page 62) procedure, do not start the Qlik Sense services.*

    a. Qlik Sense Repository Service (QRS)
    If the user running Qlik Sense services is not local administrator on the machine, you need to start Repository.exe from an elevated command prompt using the -bootstrap parameter.

See: *Qlik Sense Repository Service (page 20)*

b. Qlik Sense Proxy Service (QPS), Qlik Sense Engine Service (QES), Qlik Sense Scheduler Service (QSS), and Qlik Sense Printing Service (QPR) in no specific order

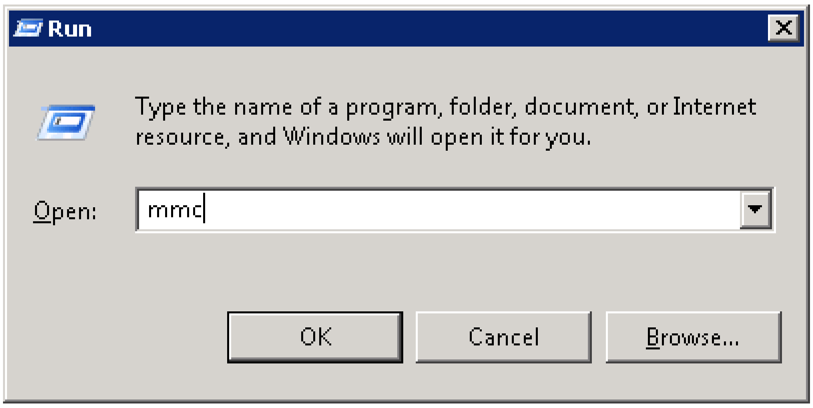
The order is important because the QRS is dependent on the QRD and the rest of the services are dependent on the QRS.

Moving a certificate

A backed up server certificate can be moved from a node in a Qlik Sense site to another node in the same site.

Proceed as follows to install the crypto key for the repository database on a new node in the site:

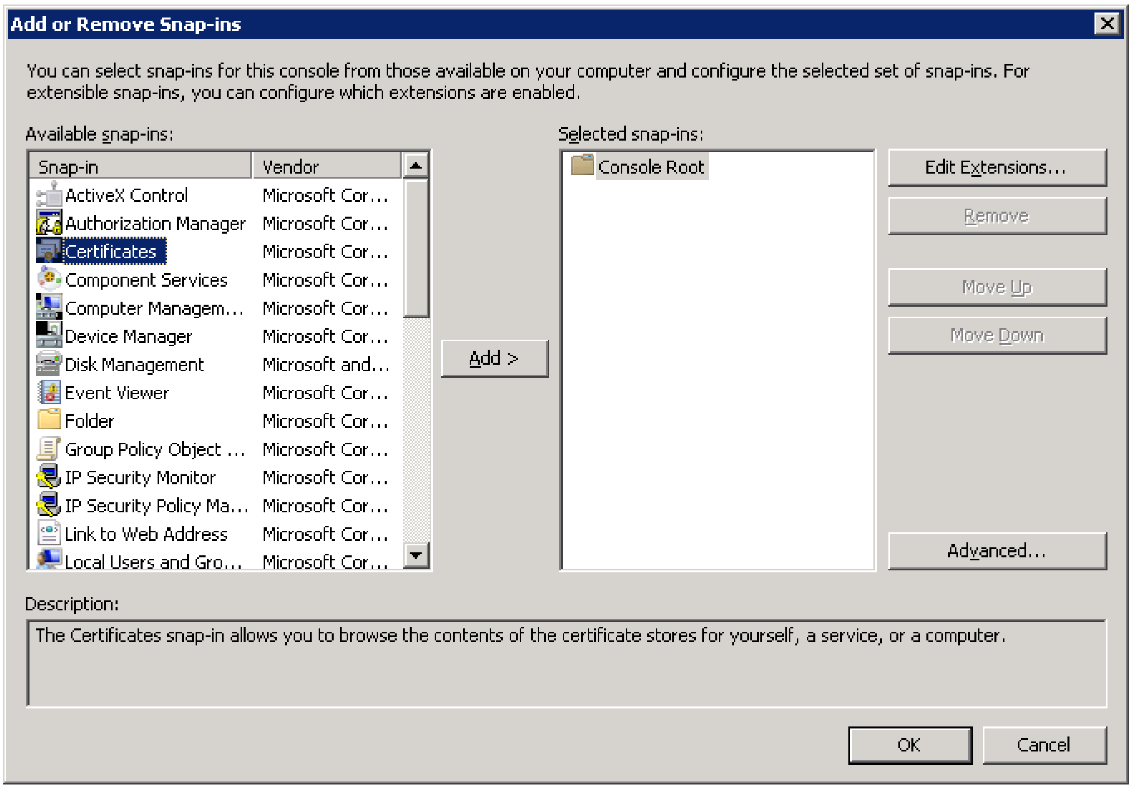
1. Open the Task Manager in Microsoft Windows and stop all Qlik Sense services except the Qlik Sense Repository Database (QRD) service.
2. Select **Start**>**Run**.
3. Enter *mmc* and click **OK**.



Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: mmc

4. Select **File**>**Add/Remove Snap-in**.
5. Double-click **Certificates**.



Add or Remove Snap-ins

You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.

Available snap-ins:

Snap-in	Vendor
ActiveX Control	Microsoft Cor...
Authorization Manager	Microsoft Cor...
Certificates	Microsoft Cor...
Component Services	Microsoft Cor...
Computer Managem...	Microsoft Cor...
Device Manager	Microsoft Cor...
Disk Management	Microsoft and...
Event Viewer	Microsoft Cor...
Folder	Microsoft Cor...
Group Policy Object ...	Microsoft Cor...
IP Security Monitor	Microsoft Cor...
IP Security Policy Ma...	Microsoft Cor...
Link to Web Address	Microsoft Cor...
Local Users and Gro...	Microsoft Cor...

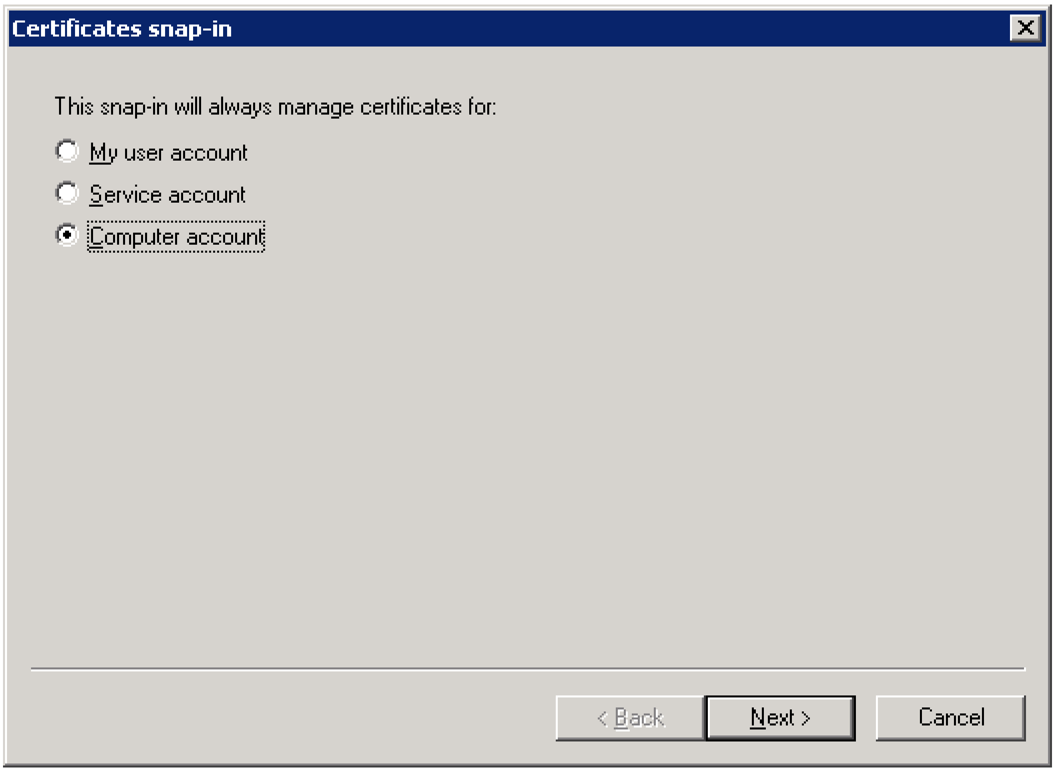
Selected snap-ins:

Console Root

Description:

The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.

6. Select **Computer account** and click **Next**.



Certificates snap-in

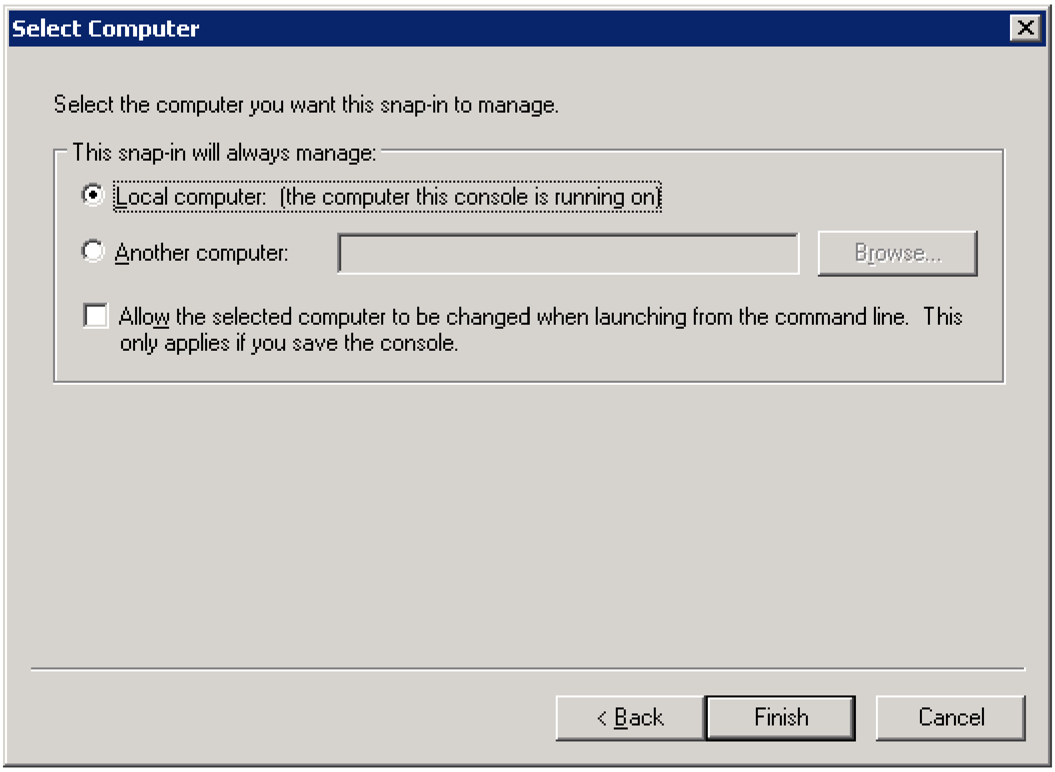
This snap-in will always manage certificates for:

○ My user account

○ Service account

● Computer account

7. Select **Local computer** and click **Finish**.



Select Computer

Select the computer you want this snap-in to manage.

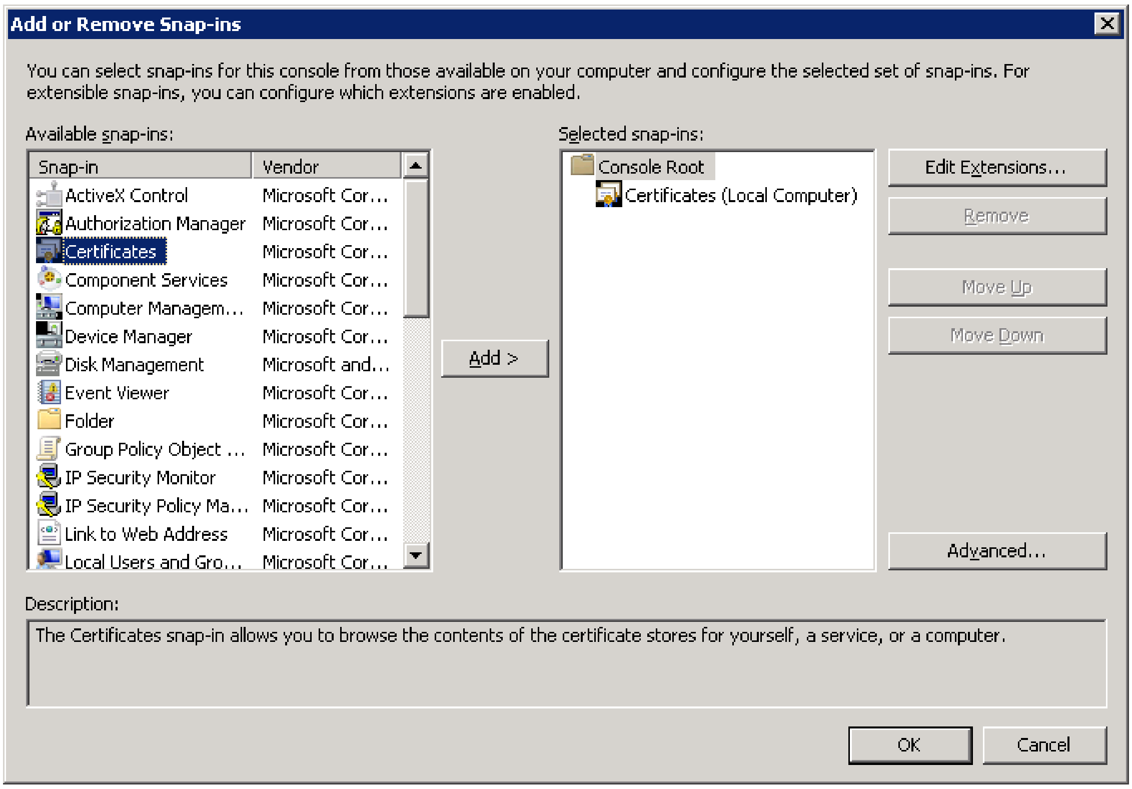
This snap-in will always manage:

● Local computer: (the computer this console is running on)

○ Another computer:

□ Allow the selected computer to be changed when launching from the command line. This only applies if you save the console.

8. Click **OK**.



Add or Remove Snap-ins

You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.

Available snap-ins:

Snap-in	Vendor
ActiveX Control	Microsoft Cor...
Authorization Manager	Microsoft Cor...
Certificates	Microsoft Cor...
Component Services	Microsoft Cor...
Computer Managem...	Microsoft Cor...
Device Manager	Microsoft Cor...
Disk Management	Microsoft and...
Event Viewer	Microsoft Cor...
Folder	Microsoft Cor...
Group Policy Object ...	Microsoft Cor...
IP Security Monitor	Microsoft Cor...
IP Security Policy Ma...	Microsoft Cor...
Link to Web Address	Microsoft Cor...
Local Users and Gro...	Microsoft Cor...

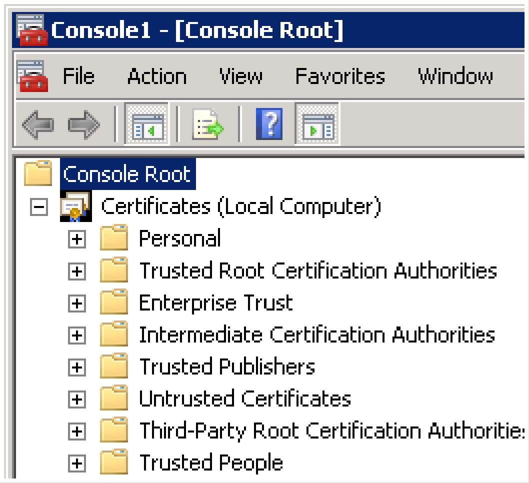
Selected snap-ins:

Console Root
Certificates (Local Computer)

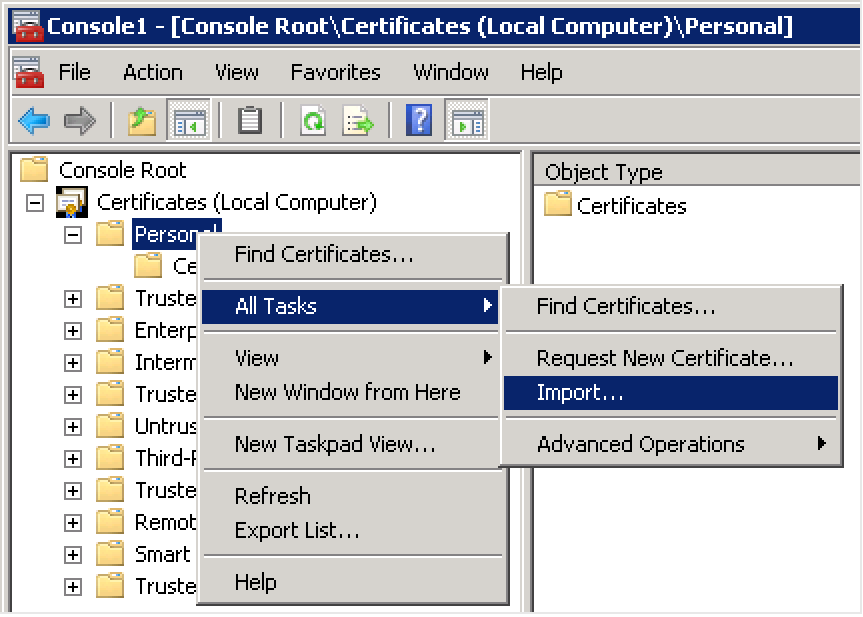
Description:

The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.

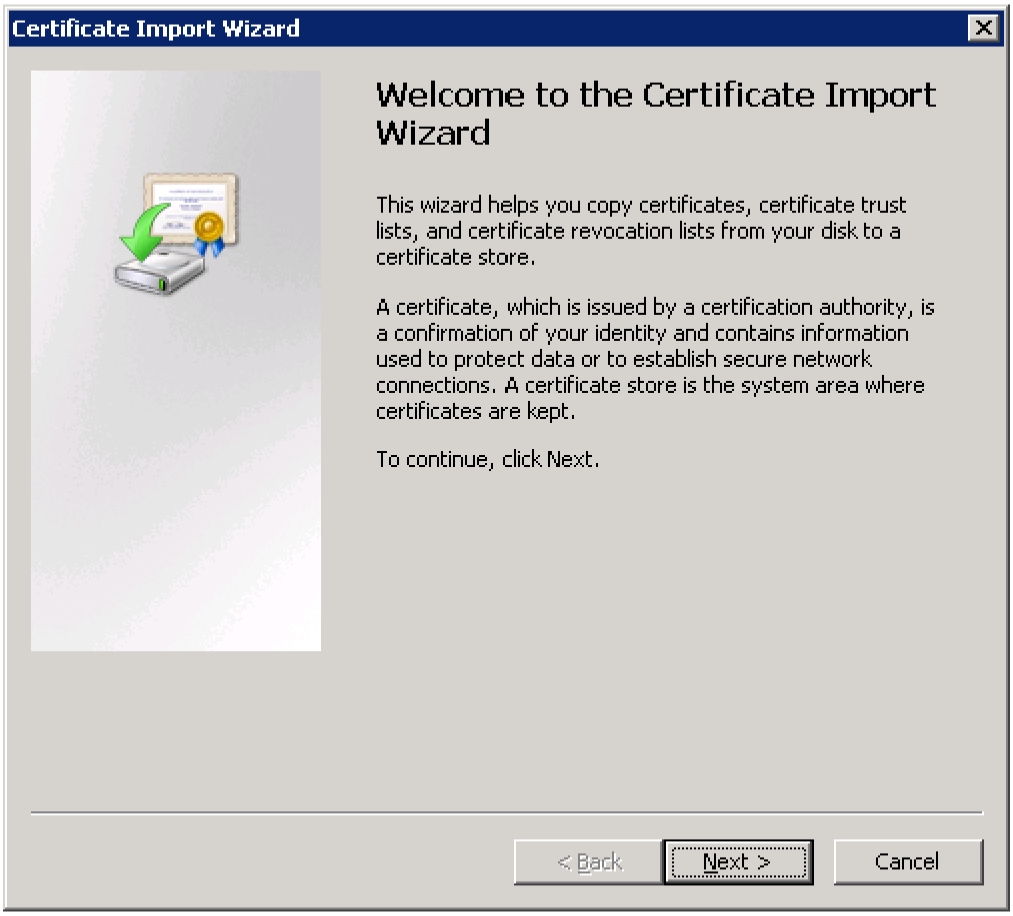
9. Expand **Certificates (Local Computer)** in the left panel.


Console1 - [Console Root]
File Action View Favorites Window
Console Root
Certificates (Local Computer)
Personal
Trusted Root Certification Authorities
Enterprise Trust
Intermediate Certification Authorities
Trusted Publishers
Untrusted Certificates
Third-Party Root Certification Authoritie:
Trusted People

10. Right-click the **Personal** folder and select **All Tasks**>**Import**.



11. Click **Next**.



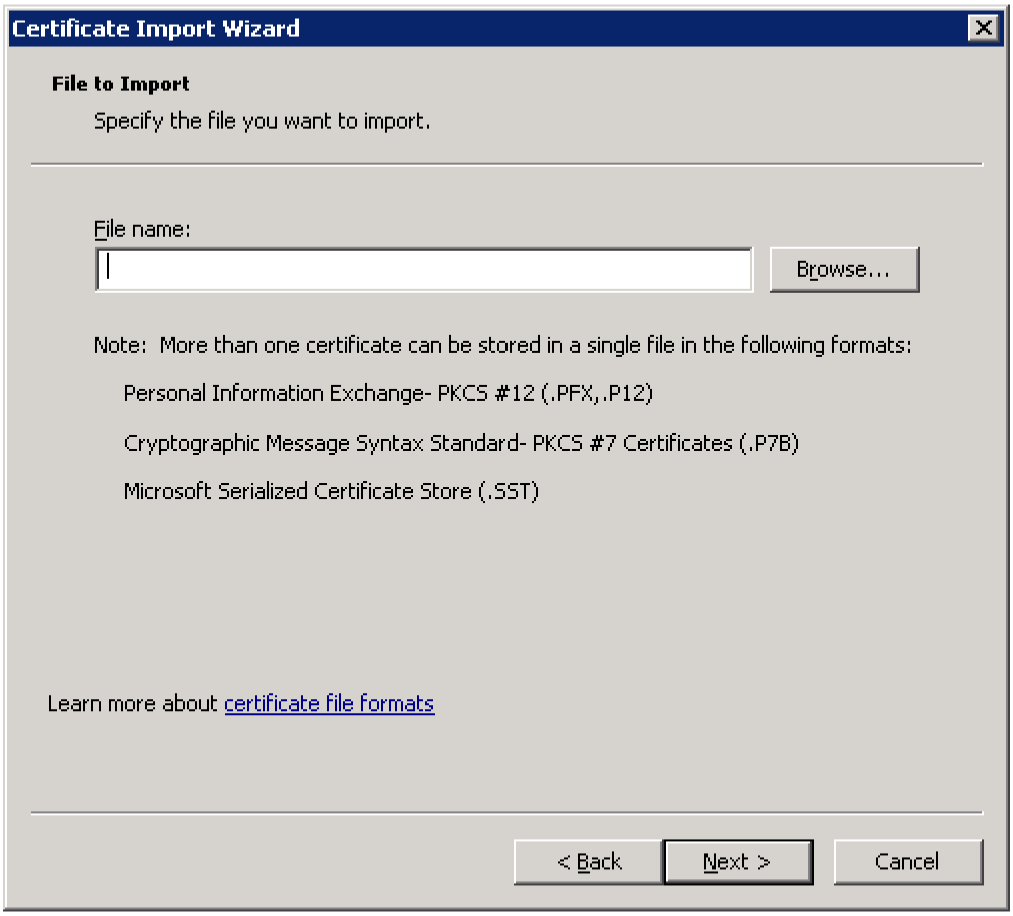
Welcome to the Certificate Import Wizard

This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

To continue, click Next.

12. Browse to the file that contains the backed up server certificate. The server certificate a) has the same name as the Domain Name System (DNS) name of the machine, and b) is signed by the CA for all nodes in the site. Then click **Next**.



Certificate Import Wizard

File to Import

Specify the file you want to import.

File name:

Note: More than one certificate can be stored in a single file in the following formats:

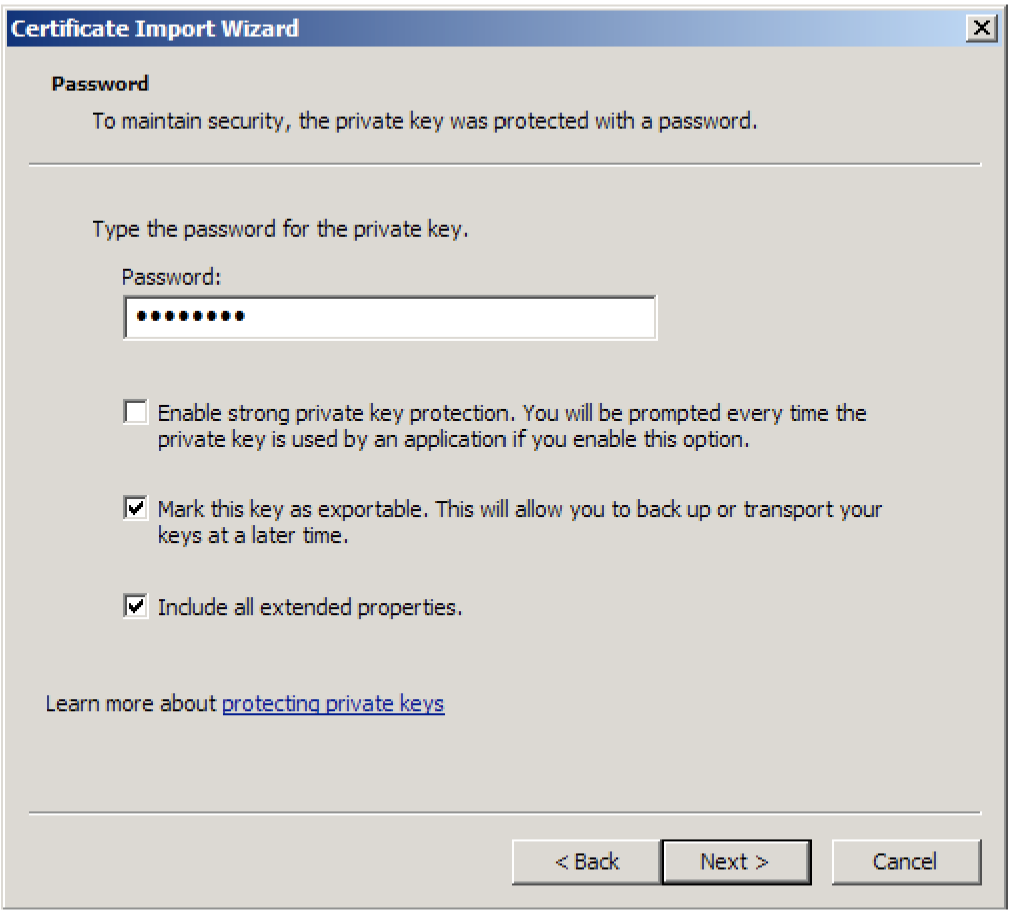
Personal Information Exchange- PKCS #12 (.PFX,.P12)

Cryptographic Message Syntax Standard- PKCS #7 Certificates (.P7B)

Microsoft Serialized Certificate Store (.SST)

Learn more about certificate file formats

13. Enter the password for the *.pfx* file (that is, the password that was given when the file was exported).
14. Select **Mark this key as exportable** and **Include all extended properties**. Then click **Next**.



Certificate Import Wizard

Password

To maintain security, the private key was protected with a password.

Type the password for the private key.

Password: ●●●●●●●●●

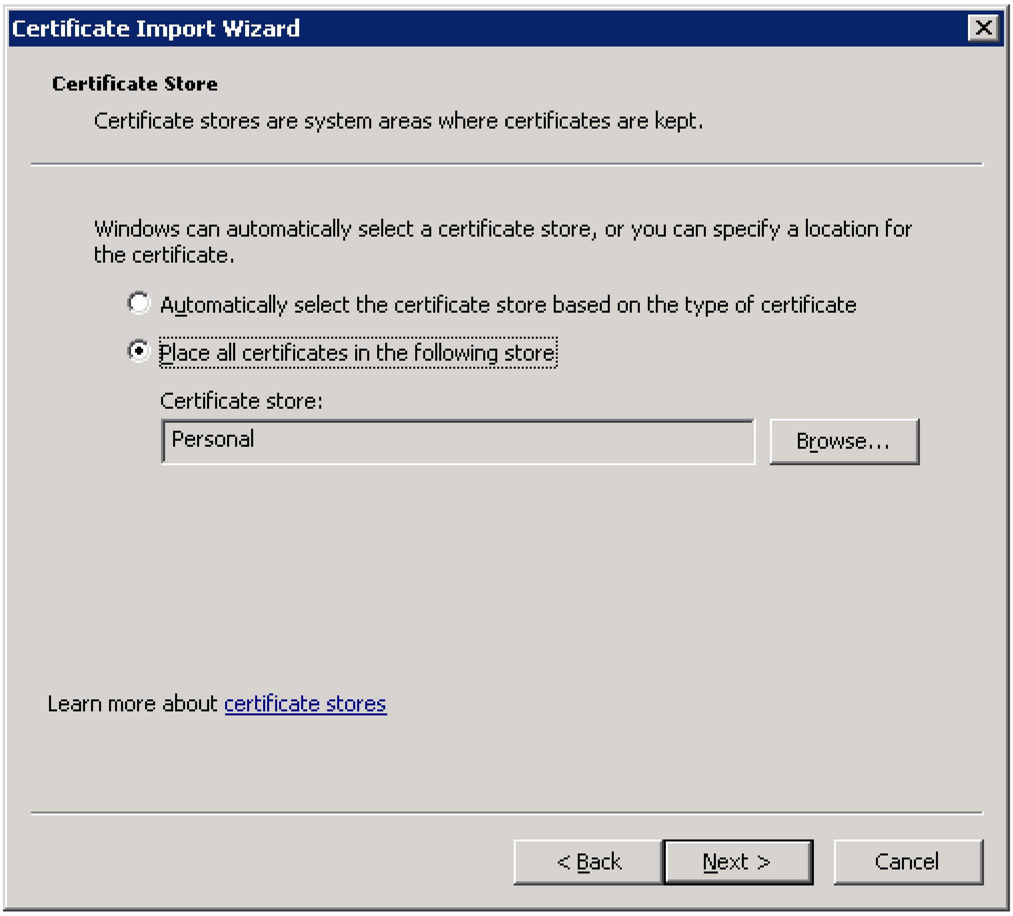
□ Enable strong private key protection. You will be prompted every time the private key is used by an application if you enable this option.

☑ Mark this key as exportable. This will allow you to back up or transport your keys at a later time.

☑ Include all extended properties.

Learn more about protecting private keys

15. Select **Place all certificates in the following store** and click **Next**.



Certificate Import Wizard

Certificate Store

Certificate stores are system areas where certificates are kept.

Windows can automatically select a certificate store, or you can specify a location for the certificate.

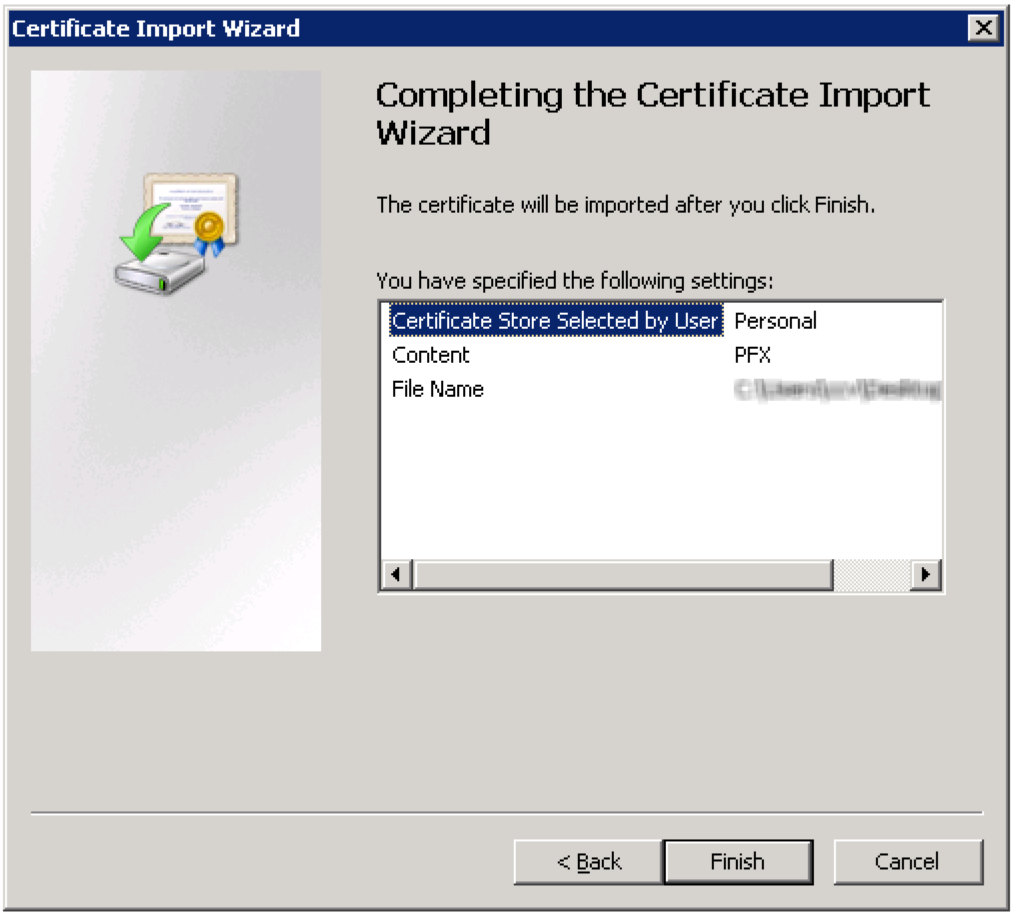
○ Automatically select the certificate store based on the type of certificate

● Place all certificates in the following store

Certificate store: Personal

Learn more about certificate stores

16. Click **Finish**.



Certificate Import Wizard

Completing the Certificate Import Wizard

The certificate will be imported after you click Finish.

You have specified the following settings:

Certificate Store Selected by User	Personal
Content	PFX
File Name	[redacted]

17. Click the **Refresh** button (⟳) and check that the imported server certificate is available in the **Personal** folder.
18. Close the MMC console.

    No changes have to be saved.

19. Start the Qlik Sense services. If the services are started manually, start them in the following order:
    a. Qlik Sense Repository Service (QRS)
    If the user running Qlik Sense services is not local administrator on the machine, you need to start Repository.exe from an elevated command prompt using the -bootstrap parameter.
    See: *Qlik Sense Repository Service (page 20)*

    b. Qlik Sense Proxy Service (QPS), Qlik Sense Engine Service (QES), Qlik Sense Scheduler Service (QSS), and Qlik Sense Printing Service (QPR) in no specific order

    The order is important because the QRS is dependent on the QRD and the rest of the services are dependent on the QRS.

See also:

🗋 *Backing up and restoring certificates (page 73)*

4 Security

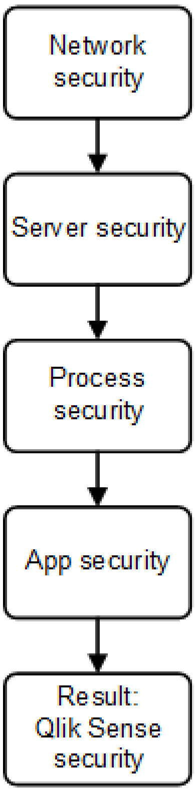
The security in Qlik Sense consists of the following parts:

- Protection of the platform: How the Qlik Sense platform itself is protected and how it communicates and operates.
- Authentication: Who is the user and how can the user prove it? Qlik Sense uses standard authentication protocols (for example, Integrated Windows Authentication), HTTP headers, and ticketing to authenticate every user requesting access to data.
- Authorization: What does the user have access to? Authorization is the procedure of granting or denying users access to resources.

4.1 Protecting the platform

The security in Qlik Sense does not depend only on the Qlik Sense software. It also relies on the security of the environment that Qlik Sense operates in. This means that the security of, for example, the operating system and the cryptographic protocols (such as TLS/SSL) has to be set up and configured to provide the security needed for Qlik Sense.

The figure below shows the components that have to be considered in order to maximize the security.



Network security

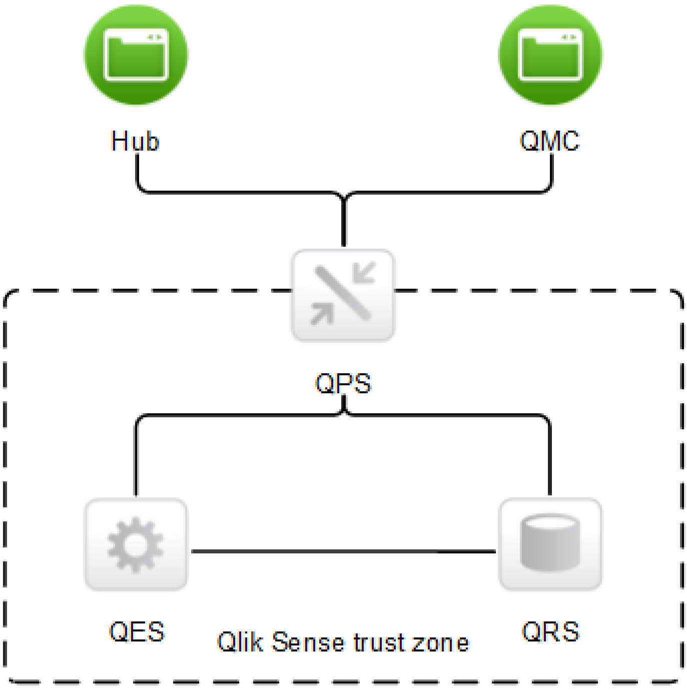
For all Qlik Sense components to communicate with each other in a secure way, they need to build trust.

In Qlik Sense, all communication between the Qlik Sense services and clients is based on web protocols. The web protocols use Transport Layer Security (TLS) for encryption and exchange of information and keys and certificates for authentication of the communicating parties.

TLS provides a way to build encrypted tunnels between identified servers or services. The parties that communicate are identified using certificates. Each tunnel needs two certificates; one to prove to the client that it is communicating with the right server and one to prove to the server that the client is allowed to communicate with the server.

So, how to make sure that the certificates are from the same Qlik Sense trust zone? All certificates that belong to a trust zone are signed with the same signature. If the signature exists in the certificate, it is accepted as proof that the certificate belongs to the trust zone.

When the protected tunnels and the correct certificates are in place, the Qlik Sense services have a trust zone to operate within. Within the trust zone, only services that belong to the specific Qlik Sense site can communicate with each other.



The Qlik Sense clients are considered to be outside of the Qlik Sense trust zone because they often run on less trusted end-user devices. The Qlik Sense Proxy Service (QPS) can bridge the two zones and allow communication between the clients and the Qlik Sense services, if the user is authenticated to the system.

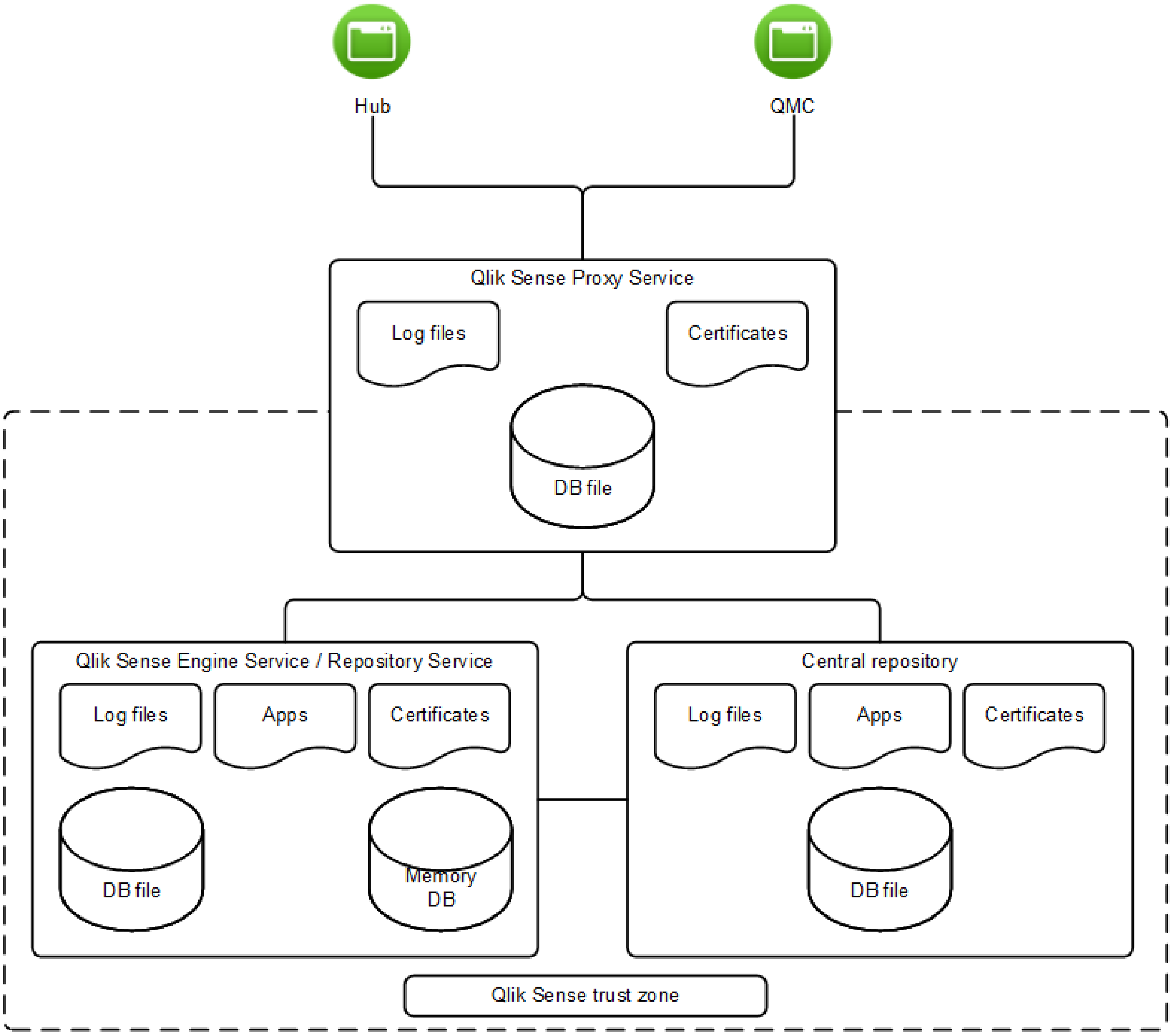
TLS-protected tunnels can be used to secure the communication between the Qlik Sense clients and the QPS. As the clients are outside of the Qlik Sense trust zone, the communication between the clients and the QPS uses a certificate with a different signature than the one used within the trust zone.

See also:

🗋 *Certificate trust (page 106)*

Server security

Qlik Sense uses the server operating system to gain access to resources. The operating system provides a security system that controls the use of the server resources (for example, storage, memory, and CPU). Qlik Sense uses the security system controls to protect its resources (for example, files, memory, processes, and certificates) on the server.



Through the use of access control, the security system grants access to Qlik Sense files (for example, log files, database files, certificates, and apps) only to certain users on the server.

The security system also protects the server memory, so that only authorized processes are allowed to write to the Qlik Sense part of the memory.

In addition, the security system is responsible for assigning users to processes. This is used to restrict who is allowed to interact with the Qlik Sense processes on the server. The processes are also restricted in terms of which parts of the operating system they are allowed to access.

So, by using the controls in the security system, a secure and protected environment can be configured for the Qlik Sense processes and files.

Process security

Each process executes in an environment that poses different threats to the process. In this layer of the security model, the focus is on ensuring that the software is robust and thoroughly analyzed from a security perspective.

Rugged software

For software to be considered as rugged, it must cope with all potential threats to the confidentiality, integrity, and availability of the information, and be robust when used in ways not anticipated.

Several mitigating actions have been implemented in the Qlik Sense software in order to make it rugged:

- Authorization of communication using certificates
- Validation of all external data that is sent to the system
- Encoding of content to avoid injection of malicious code
- Use of protected memory
- Encryption of data
- Audit logging
- Use of checksums
- Isolated execution of external components
- Escaping of SQL data

Threat analysis

To ensure that the Qlik Sense software is secure and rugged, threat analysis of the design has been performed as part of the development process. The following threat areas, often abbreviated as STRIDE, have been covered:

- **S**poofing
- **T**ampering
- **R**epudiation
- **I**nformation disclosure
- **D**enial of service
- **E**levation of privilege

In addition to the threat analyses, exploratory security testing has also been performed on the Qlik Sense software.

App security

The major components of the Qlik Sense app security are:

- Access control system: The access control system grants users access to the resources in Qlik Sense.
- Data reduction: The data reduction functionality is a way to dynamically change which data a user can view. This makes it possible to build apps that can be consumed by many users, but with different data sets that are dynamically created based on user information. The reduction of data is performed by the Qlik Sense Engine Service (QES).

Using these components, the resources and data (that is, the content) consumed by the Qlik Sense users can be secured.

4.2 Authentication

All authentication in Qlik Sense is managed by the Qlik Sense Proxy Service (QPS). The QPS authenticates all users regardless of Qlik Sense client type. This means that the QPS also authenticates users of the Qlik Management Console (QMC).

> *In Qlik Sense, authentication and authorization are two distinct, unconnected actions. In addition, the sources of information used for authentication do not have to be the same as for authorization, and the other way around.*

Qlik Sense always asks an external system to verify who the user is and if the user can prove it. The interaction between Qlik Sense and the external identity provider is handled by authentication modules.

For a module to communicate with Qlik Sense, it has to be trusted. Transport Layer Security (TLS) and certificate authentication are used to authorize external components for communication with Qlik Sense.

In Qlik Sense, the authentication of a user consists of three distinct steps:

1. Authentication module: Get the user identity and credentials.
2. Authentication module: Request an external system to verify the user identity using the credentials.
3. Transfer the user to Qlik Sense using the Ticket API, the Session API, headers, or SAML.

The first two steps are always handled by the authentication module. It is up to the authentication module to verify the user in an appropriate way.

The third step can be performed in the following ways:

- Using the Ticket API, which transfers the user and the user's properties using a one-time ticket.
- Using the Session API, whereby an external module can transfer web sessions that identify the user and the user's properties to Qlik Sense.
- Using headers, with which a trusted system can transfer the user using HTTP headers. This is a common solution for integrating with Single Sign-On (SSO) systems.
- Qlik Sense can be configured to allow anonymous users (using, for example, SAML).

See also:

Network security (page 101)

Default authentication module

After a default installation of Qlik Sense, the Qlik Sense Proxy Service (QPS) includes a module that handles authentication of Microsoft Windows users. The module supports the use of Kerberos and NTLM.

If you want to use Kerberos authentication, you need to make sure that browsers that are used to access the Qlik Sense are configured to support Kerberos.

> *The default authentication module requires that the proxy that handles the authentication is part of the Microsoft Windows domain.*

Certificate trust

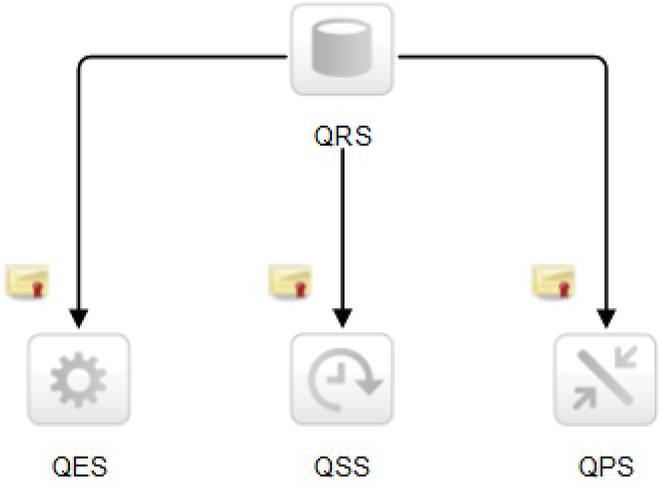
Qlik Sense uses certificates for authentication. A certificate provides trust between nodes within a site.

This section describes how to deploy certificates for use in Qlik Sense.

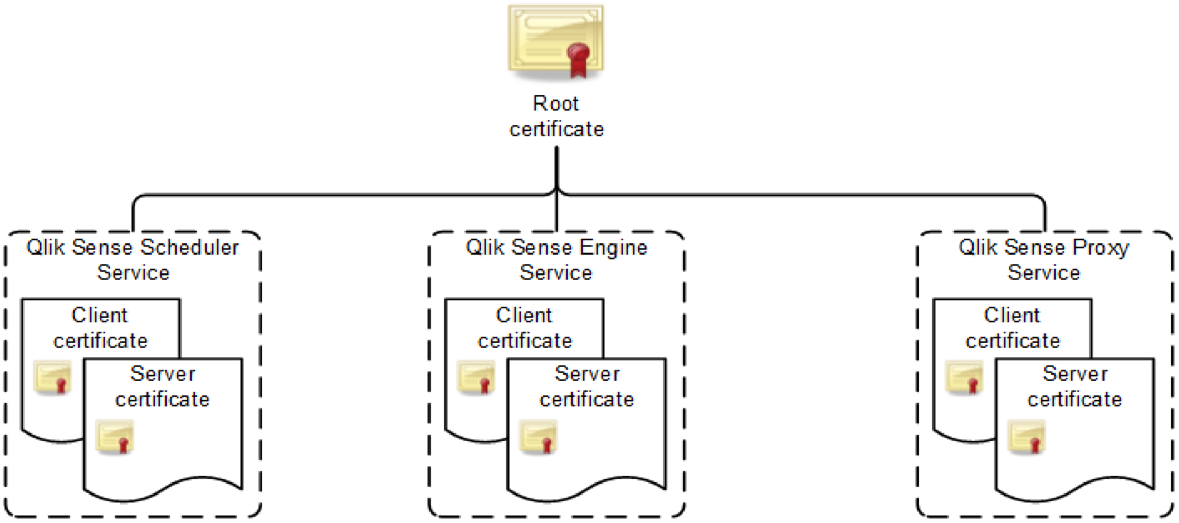
Architecture

Certificates are used within a Qlik Sense site to authenticate communication between services that reside on different nodes. In addition, certificates can be used to build a trust domain between services that are located in different domains or areas (for example, internal networks, extranets, and Internet) without having to share a Microsoft Active Directory (AD) or other user directories.

The architecture is based on the master Qlik Sense Repository Service (QRS) on the central node acting as the certificate manager or Certificate Authority (CA). The master QRS creates and distributes certificates to all nodes within a site. The master QRS is therefore an important part of the security solution and has to be managed from a secure location to keep the certificate solution secure.



The root certificate for the installation is stored on the central node in the site, where the master QRS runs. All nodes with Qlik Sense services that are to be used within the site receive certificates signed with the root certificate when added to the master QRS. The master QRS (that is, the CA) issues digital certificates that contain keys and the identity of the owner. The private key is not made publicly available – it is kept secret by the nodes. The certificate enables the services in a Qlik Sense deployment to validate the authenticity of the other services. This means that the master QRS is responsible for making sure that a service that is deployed on a node is a service within the site.



After the nodes have received certificates, the communication between the Qlik Sense services is encrypted using Transport Layer Security (TLS) encryption.

Requirements

The requirements described in this section must be fulfilled for the certificate trust to function properly.

General

When using Transport Layer Security (TLS) in Microsoft Windows environments, the private key must be stored together with the certificate in the Windows certificate store. In addition, the account that is used to run the Qlik Sense services must have permission to access the certificate private key.

You should also make sure that TLS 1.0 is enabled in Windows.

If you want to use TLS 1.2 authentication, you need to enable TLS 1.2 support in the Windows registry of the server machine. You should consider the impact of enabling TLS 1.2, as this is a global system setting.

Communication ports

To set up certificate trust, the Qlik Sense Repository Services (QRSs) require that the ports listed in the following table can be opened and used for communication. If any communication passes through a network firewall, the ports in the firewall must be opened and configured for the services.

Port no.	Description
4570	Certificate password verification port, only used within multi-node sites by Qlik Sense Repository Services (QRSs) on rim nodes to receive the password that unlocks a distributed certificate. The port can only be accessed from localhost and it is closed immediately after the certificate has been unlocked. The communication is always unencrypted. This port uses HTTP for communication.
4444	Security distribution port, only used by Qlik Sense Repository Services (QRSs) on rim nodes to receive a certificate from the master QRS on the central node. The communication is always unencrypted, but the transferred certificate package is password-protected. This port uses HTTP for communication.

See: *Ports (page 26)*

Unlocking distributed certificates

When adding a new rim node to a site, the distributed certificate needs to be unlocked.

See: Manage Qlik Sense sites

Confirming certificates using Microsoft Management Console

Certificates can be visually confirmed in the Microsoft Management Console (MMC) with the certificate snap-in added.

If the certificates have been properly deployed, they are available in the locations listed in the table.

Certificate	Location
QlikClient	**Certificates - Current User**>**Personal**>**Certificates**
<full computer name>-CA	**Certificates - Current User**>**Trusted Root Certification Authorities**>**Certificates**
<full computer name>-CA	**Certificates (Local Computer)**>**Trusted Root Certification Authorities**>**Certificates**
<computer name>	**Certificates (Local Computer)**>**Personal**>**Certificates**

Handling of certificates when a service starts

This section describes how the certificates are handled when a Qlik Sense service starts.

Client certificate

This section describes how the master Qlik Sense Repository Service (QRS) on the central node in a site handles the client certificate when a Qlik Sense service starts.

The client certificate is located in the following place in the Microsoft Windows certificate store:

Current User>Personal>Certificates

When a Qlik Sense service starts, the QRS searches the certificate store to see if there are any Qlik Sense certificates. Depending on the results of the search, the QRS does the following:

- If no client certificate is found, the QRS creates a new certificate.
- If only one client certificate is found, the QRS checks if it is valid. If the certificate is not valid, the QRS deletes the certificate and creates a new one. In addition, the QRS logs that an invalid certificate was found and deleted.
- If more than one client certificate is found, the QRS deletes all certificates and creates a new one. Duplicates are not allowed. In addition, the QRS logs the number of valid and invalid certificates that were found and deleted.

Server certificate

This section describes how the master Qlik Sense Repository Service (QRS) on the central node in a site handles the server certificate when a Qlik Sense service starts.

The server certificate is located in the following place in the Microsoft Windows certificate store:

Local Computer>Personal>Certificates

When a Qlik Sense service starts, the QRS searches the certificate store to see if there are any Qlik Sense certificates. Depending on the results of the search, the QRS does the following:

- If no server certificate is found, the QRS creates a new certificate.
- If only one server certificate is found, the QRS checks if it is valid. If the certificate is not valid, the QRS deletes the certificate and creates a new one. In addition, the QRS logs that an invalid certificate was found and deleted.
- If more than one server certificate is found, the QRS deletes all certificates and creates a new one. Duplicates are not allowed. In addition, the QRS logs the number of valid and invalid certificates that were found and deleted.

Root certificate

This section describes how the master Qlik Sense Repository Service (QRS) on the central node in a site handles the root certificate when a Qlik Sense service starts.

The root certificate is located in the following places in the Microsoft Windows certificate store:

Current User>Trusted Root Certification Authorities>Certificates

Local Computer>Trusted Root Certification Authorities>Certificates

When a Qlik Sense service starts, the QRS searches the certificate store to see if there are any Qlik Sense certificates. Depending on the results of the search, the QRS does the following:

- If no root certificate is found, the QRS creates a new certificate.
- If only one root certificate is found, the QRS checks if it is valid. If it is not valid, the QRS logs a fatal error that an invalid root certificate was found, which means that the service is shut down and that the

administrator manually has to delete any unwanted certificates. In addition, the QRS logs information on the certificates that are affected by this.

- If more than one root certificate is found, the QRS logs a fatal error that an invalid root certificate was found, which means that the service is shut down and that the administrator manually has to delete any unwanted certificates. In addition, the QRS logs information on the certificates that are affected by this.

> *In order not to break any certificate trust between machines, the QRS does not remove any root certificates. It is up to the administrator to decide on what to do with invalid root certificates.*

See also:

Services (page 17)

Definition of invalid certificate

The definition of an invalid certificate is as follows:

- The operating system considers the certificate to be too old or the certificate chain is incorrect or incomplete.
- The Qlik Sense certificate extension (OID “1.3.6.1.5.5.7.13.3”) is missing or does not reflect the location of the certificate:
  - Current User/Personal certificate location: Client
  - Local Machine/Personal certificate location: Server
  - Local Machine/Trusted Root certificate location: Root
  - Current User/Trusted Root certificate location: Root
- The server, client, and root certificates on the central node do not have a private key that the operating system allows them to access.
- The server and client certificates are not signed by the root certificate on the machine.

Maximum number of trusted root certificates

When a Qlik Sense service starts, it checks the number of trusted root certificates on the machine where it is running. If there are more that 300 certificates on the machine, warning messages containing the following information are logged:

- There are too many root certificates for the service to trust.
- The Microsoft Windows operating system will truncate the list of certificates during the Transport Layer Security (TLS) handshake.

If the Qlik Sense root certificate (<*host-machine*>*-CA*) that the Qlik Sense client certificate belongs to is deleted from the list of certificates because of the truncation, the service cannot be authenticated.

To manually view the root certificates on a machine, open the Microsoft Management Console (MMC) and go to **Certificates (Local Computer)**>**Trusted Root Certification Authorities**.

Authentication solutions

This section describes various authentication solutions for Qlik Sense.

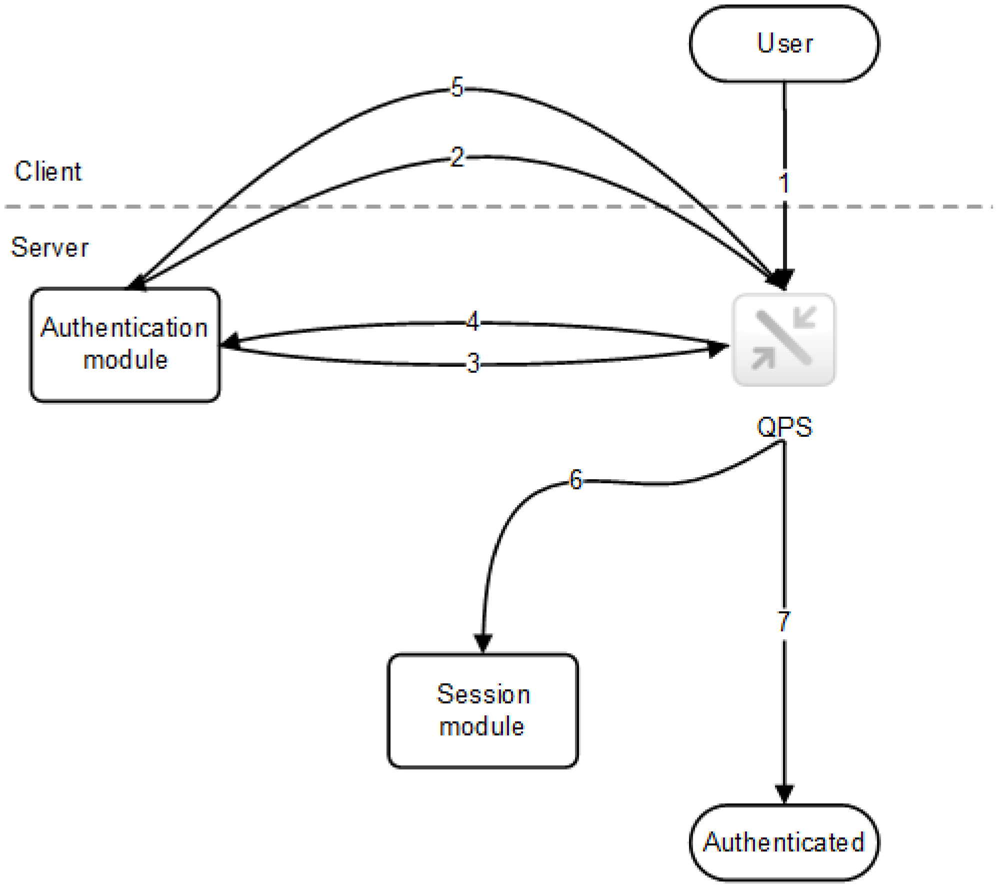
Ticket solution

The ticket solution is similar to a normal ticket. The user receives a ticket after having been verified. The user then brings the ticket to Qlik Sense and, if the ticket is valid, is authenticated. In order to keep the tickets secure, the following restrictions apply:

- A ticket is only valid for a short period of time.
- A ticket is only valid once.
- A ticket is random and therefore hard to guess.

All communication between the authentication module and the Qlik Sense Proxy Service (QPS) uses Transport Layer Security (TLS) and must be authorized using certificates.

The figure below shows a typical flow for authenticating a user with tickets.



1. The user accesses Qlik Sense.
2. Qlik Sense redirects the user to the authentication module. The authentication module verifies the user identity and credentials with an identity provider.

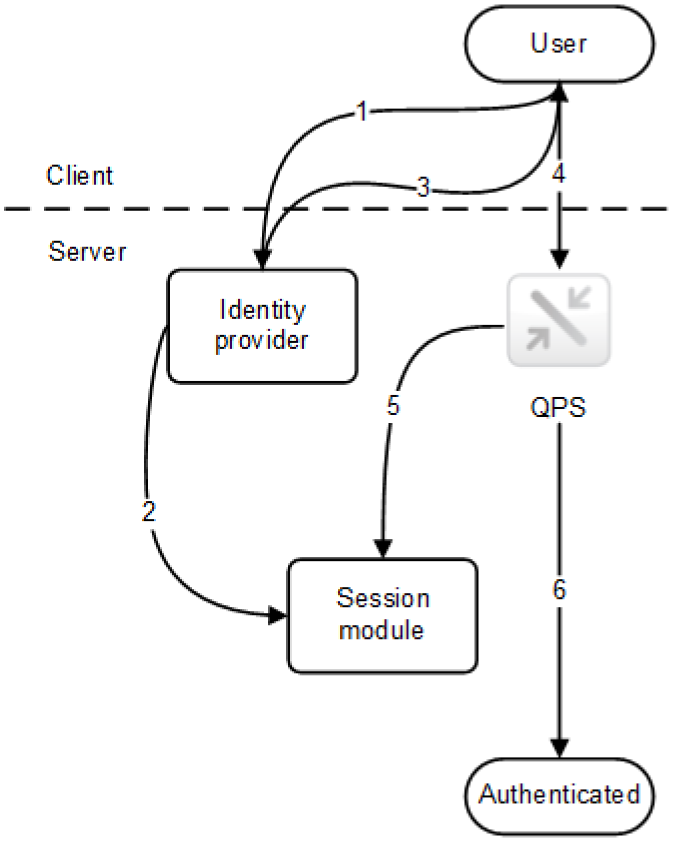
3. Once the credentials have been verified, a ticket is requested from the QPS. Additional properties may be supplied in the request.
4. The authentication module receives a ticket.
5. The user is redirected back to the QPS with the ticket. The QPS checks that the ticket is valid and has not timed out.
6. A proxy session is created for the user.
7. The user is now authenticated.

Session solution

The session solution allows the Qlik Sense Proxy Service (QPS) to use a session from an external system to validate who the user is.

All communication between the authentication module and the QPS uses Transport Layer Security (TLS) and must be authorized using certificates.

The figure below shows a typical flow for authenticating a user using a session from an external system.



1. The user accesses the identity provider, which, for example, can be integrated into a portal. The identity provider gets the user identity and credentials and then verifies them. After that, the identity provider creates a new session.
2. The identity provider registers the session token with the Qlik Sense session module.
3. The identity provider sets the session token as a session cookie.

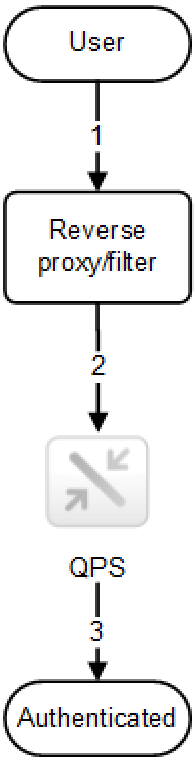
4. The user accesses the QPS to get content (for example, through an iframe in the portal).
5. The QPS validates the session to the session module.
6. If the session is valid and has not yet timed out, the user is authenticated.

The name of the session cookie used by the authentication module can be configured in the Qlik Management Console (QMC).

Header solution

Header authentication is often used in conjunction with a Single Sign-On (SSO) system that supplies a reverse proxy or filter for authenticating the user.

The figure below shows a typical flow for authenticating a user using header authentication.



1. The user accesses the system and authenticates to the reverse proxy.
2. The reverse proxy injects the username into a defined HTTP header. The header must be included in every request to the Qlik Sense Proxy Service (QPS).
3. The user is authenticated.

For this solution to be secure, the end-user must not be able to communicate directly with the QPS but instead be forced to go through the reverse proxy/filter.

The reverse proxy/filter must be configured to preserve the host name, that is, the host header from the client must not be modified by the reverse proxy/filter.

The name of the HTTP header used for the user can be configured in the Qlik Management Console (QMC).

SAML

Security Assertion Markup Language (SAML) is an XML-based, open-standard data format for exchanging authentication and authorization data between parties (for example, between an identity provider and a service provider). SAML is typically used for web browser Single Sign-On (SSO).

How SAML works

The SAML specification defines three roles:

- Principal: Typically a user
- IdP: The identity provider
- SP: The service provider

The principal requests a service from the SP, which requests and obtains an identity assertion from the IdP. Based on the assertion, the SP decides whether or not to perform the service requested by the principal.

SAML in Qlik Sense

Qlik Sense supports SAML V2.0 by:

- Implementing an SP that can integrate with external IdPs
- Supporting HTTP Redirect Binding and HTTP POST Binding
- Supporting SAML properties for access control of resources and data

Limitations:

- Qlik Sense does not support SAML message signature validation.

Anonymous users

If anonymous use of Qlik Sense is allowed, users who are not authenticated are not automatically redirected to an authentication module. Instead, the user first gets anonymous access and is then, if the user chooses to sign in, redirected to the authentication module to supply user identity and credentials.

4.3 Authorization

Authorization is the procedure of granting or denying users access to resources.

In Qlik Sense, authentication and authorization are two distinct, unconnected actions. In addition, the sources of information used for authentication do not have to be the same as for authorization, and the other way around.

In Qlik Sense, there are two authorization systems:

- Access control: The access control system grants users access to the resources in Qlik Sense. The access control system is implemented in the Qlik Sense Repository Service (QRS) and independent of the operating system.
- Data reduction: The data reduction functionality is a way to dynamically change which data a user can view. This makes it possible to build apps that can be consumed by many users, but with different data sets that are dynamically created based on user information. The reduction of data is performed by the Qlik Sense Engine Service (QES).

The two authorization systems are unconnected and configured separately.

Access control

This section describes the different types of access control:

- Resource access control: Is the user allowed to access the app? Which functions in the app is the user allowed to use (for example, printing, exporting, and snapshots)?
- Administrator access control: Which access rights are needed for the different roles and responsibilities of the administrators?

Resource access control

The resource access control system in Qlik Sense is based on properties. This means that the access is based on rules that refer to properties connected to resources and users in Qlik Sense.

All authorization to resources is enforced by the Qlik Sense Repository Service (QRS). The QRS only gives other Qlik Sense services access to resources that the current user is allowed to access.

The resource access control system determines the access based on the following parameters:

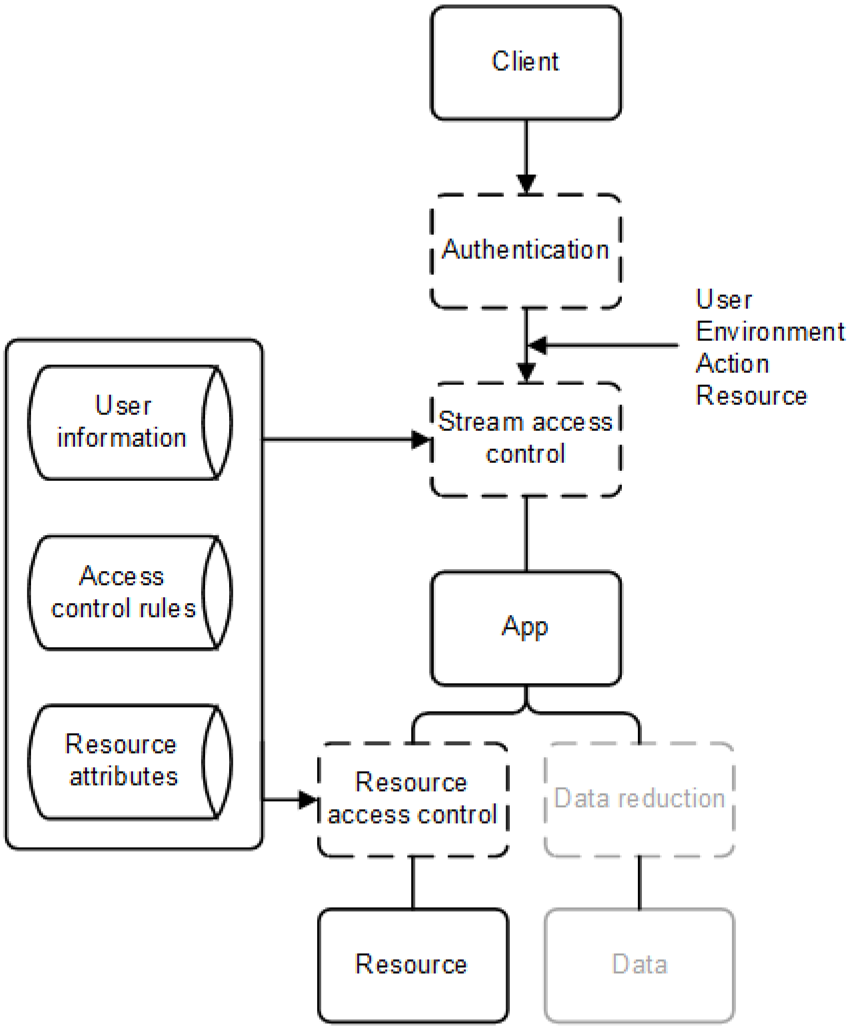
- User name and user properties: The user name and user properties are supplied by the Qlik Sense Proxy Service (QPS) that authenticated the user.
- Action: The method that the user is trying to perform on a resource (for example, create, read, or print).
- Resource: The entity that the user is trying to perform an action on (for example, app, sheet, or object).
- Environment: The environment is supplied by the QPS and describes, for example, time, location, protection, and the type of Qlik Sense client used.

Rules

The system administrator can set up rules for the resources access control. The rules are divided into three parts:

- Resource filter: The resources that the rule applies to.
- Condition: A logical condition that, if evaluated as true, grants access.
- Action: The action that the user is allowed to perform, if the condition is true.

Properties connected to resources or users may be used in the rules. Examples of properties include the name of user or resource, type of resource, and Active Directory groups for users or custom-defined properties.



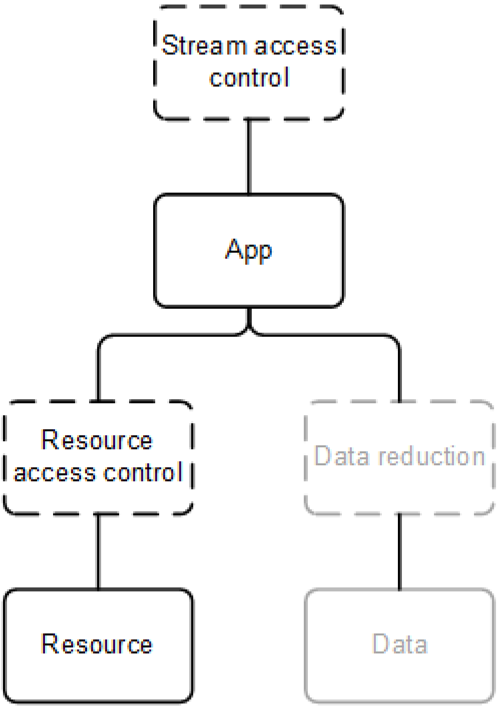
Streams

To make the management of the Qlik Sense authorization systems efficient, apps can be grouped into streams. From an authorization perspective, a stream is a grouping of apps that a group of users has read (often referred to as “subscription”) or publish access to.

By default, Qlik Sense includes the following streams:

- Everyone: All users have read and publish rights to this stream.
- Monitoring apps: Contains a number of apps for monitoring of Qlik Sense.

Streams are created and managed in the Qlik Management Console (QMC).



Administrator access control

In addition to setting up the access control for the users, it is important to configure the access control for the administrators so that they get access rights in the Qlik Management Console (QMC) that correspond to their roles and responsibilities.

Common administrator roles include:

- RootAdmin: Full access to all Qlik Sense resources.
- AuditAdmin: Read access to all resources.
- ContentAdmin: Full access to all resources except nodes, engines, repositories, schedulers, and syncs.
- DeploymentAdmin: Full access to apps, tasks, licenses, nodes, repositories, schedulers, proxies, virtual proxies, and engines.
- SecurityAdmin: Same as ContentAdmin, but with full access to proxies and virtual proxies and no access to tasks.

Data reduction

Data reduction is used to determine which data a user is allowed to see: all of it or just parts of it?

The data reduction functionality is a way to dynamically change which data a user can view. This makes it possible to build apps that can be consumed by many users, but with different data sets that are dynamically created based on user information. The reduction of data is performed by the Qlik Sense Engine Service (QES).

The definition of access rights for section access is maintained in the apps and configured through the load script.

4.4 Security summary

This section provides a summary of the Qlik Sense security system.

Authentication

Qlik Sense supports authentication in the following ways:

- The users are authenticated by the Qlik Sense Proxy Service (QPS).
- The QPS supports the use of multiple proxies and each proxy can use multiple authentication methods over a network protected by Transport Layer Security (TLS).

Authorization

Qlik Sense supports authorization in the following ways:

- Inter-server communication is authorized through Transport Layer Security (TLS) using certificates for authentication.
- The Qlik Sense Repository Service (QRS) provides property-based access control of user content.
- Authorization to data is managed using section access.

Auditing

Qlik Sense supports auditing in the following ways:

- The repository database stores information about when the database was last changed and who made the change.
- The logging framework provides audit and security logs.
- The logs are centrally stored.
- The log format is resistant to injection from the Qlik Sense clients.
- The license logs are signed with a signature to protect them from tampering.

Confidentiality

Qlik Sense supports confidentiality in the following ways:

- The network uses Transport Layer Security (TLS) for encryption and certificates for authentication.
- The locally stored information on a node, including Qlik Sense content, is protected by the operating system using server access control and file system controls.
- The process memory and loaded data for Qlik Sense are protected by the physical server and the operating system controls.
- The apps are secured using access control on the resource level.
- Sensitive information (for example, passwords and connection strings) that is used to access external data sources is stored with encryption.
- The app data is protected using data reduction.

Integrity

Qlik Sense supports integrity in the following ways:

- Stored data is protected using the operating system controls (for example, the file system).
- Sensitive information (for example, passwords and connection strings) that is used to access external data sources is stored with encryption.
- Qlik Sense does not support write back to the source system (that is, the Qlik Sense clients cannot edit the data sources).

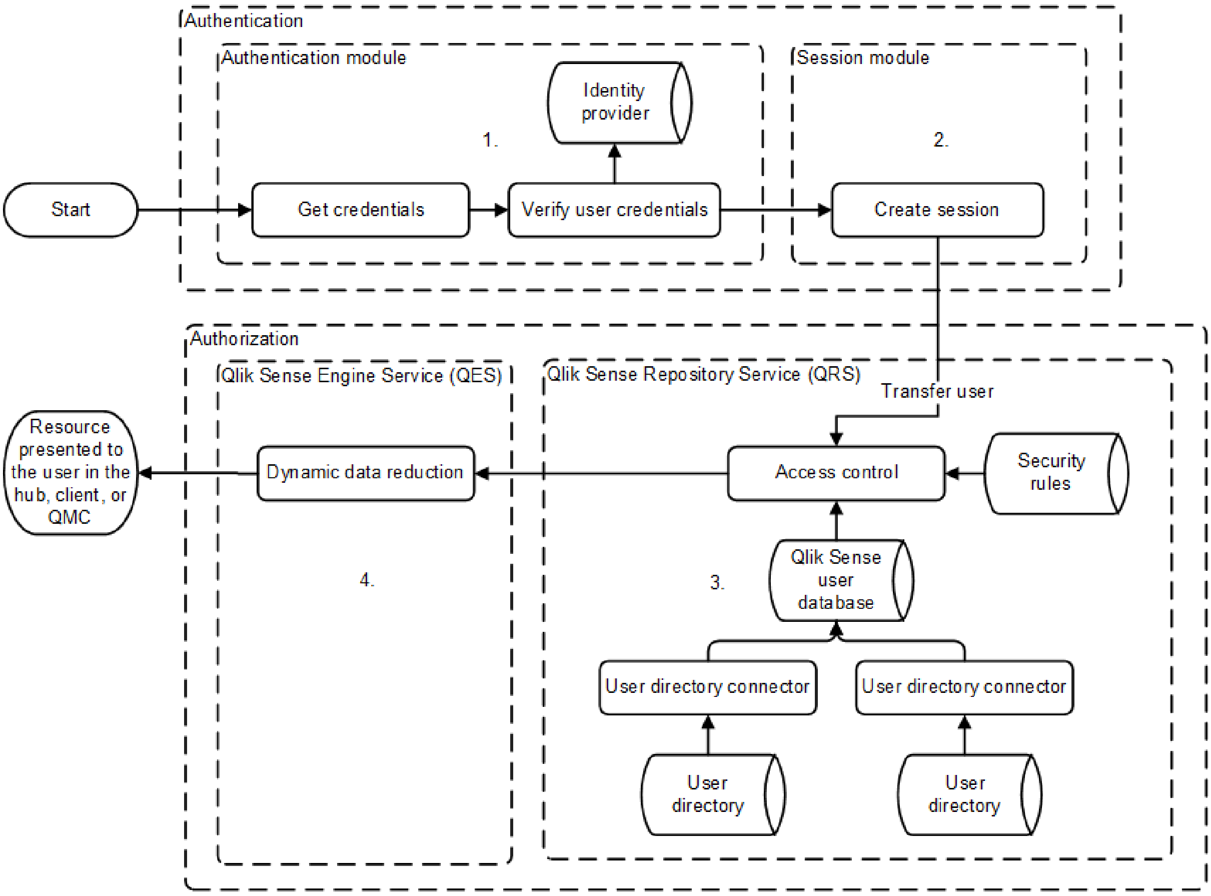
Availability

Qlik Sense supports availability in the following ways:

- The nodes in a multi-node site are resilient by design. Each node has a local copy of the data that it needs to fulfill its role.
- The Qlik Sense protocols are designed to be fault tolerant.

Security example: Opening an app

The figure below shows the flow in the Qlik Sense security system when a user logs in and opens an app.



1. Authentication: The authentication module in the Qlik Sense Proxy Service (QPS) handles the authentication. The credentials provided by the user are verified against information from the identity provider (for example, a directory service such as Microsoft Active Directory).
2. Session creation: When the user credentials have been successfully verified by the authentication module, a session is created for the user by the session module in the QPS.
3. Access control system: When the user tries to open an app, the Qlik Sense Engine Service (QES) requests the Qlik Sense Repository Service (QRS) to check if the user is authorized to perform the action. The QRS then checks the repository database, where, among other things, all users and access rights are stored.

   The users are imported into the repository database from one or more User Directories (UDs) (for example, Microsoft Active Directory) using Qlik Sense User Directory Connectors (UDCs). The import is triggered by the Qlik Sense Scheduler Service (QSS) and the intervals in-between imports can be scheduled.
4. Dynamic data reduction: When the user has been successfully authorized by the QRS, the app is opened. Before the data is displayed to the user, the QES performs a dynamic data reduction, where the data that the user is allowed to see is prepared.

See also:

🗋 *App security (page 104)*

5 Logging

The log messages produced by Qlik Sense provide important information that can be used to detect security incidents, operational problems, and policy violations.

The logging is based on the log4net component in Apache Logging Services. This means that Qlik Sense uses a standardized logging framework and conforms to standard logging procedures.

5.1 New logging framework

A new logging framework was introduced in Qlik Sense version 2.0. Unless otherwise stated, the documentation describes the new logging framework.

5.2 Legacy logging framework

The legacy logging framework is still available in Qlik Sense, but the logs are as of Qlik Sense version 2.0 referred to as trace logs. The log files remain the same, but they are stored in a new location.

See: *Trace logs (page 137)*

5.3 Reading and analyzing log files in Qlik Sense

The log files can be read and analyzed using Qlik Sense, which includes the following pre-defined, log-related data connections after installation:

- ServerLogFolder: Links to the active log files.
- ArchivedLogsFolder: Links to the archived log files.

The data connections can be edited in the Qlik Management Console (QMC).

In addition, users with root, security, content, or deployment administrator rights can use the Qlik Sense log data in apps by selecting one of the data connections listed above in the data load editor.

See also:

⇨ Apache Logging Services

5.4 Requirements

The requirements described in this section must be fulfilled for the Qlik Sense logging to function properly.

Securing the file system

The system administrator must secure the file system so that the log files cannot be tampered with.

By default, the account used for the Qlik Sense installation gets full permissions for the log folder, %ProgramData%\Qlik\Sense\Log, whereas the Users group only gets read permission. No other accounts or users get any permissions for the log folder.

Synchronizing time

The nodes within a Qlik Sense site must have synchronized time.

For the date and time stamps to be correct, all nodes within a site must be configured to synchronize their system clocks with either an internal or an external Network Time Protocol (NTP) service to ensure that all log entries are time-stamped accurately. NTP is a networking protocol for synchronizing the clocks of computer systems over packet-switched, variable-latency data networks.

Setting time zone

It is recommended that every node within a Qlik Sense site is set to the correct time zone so that the time zone corresponds to the geographical location of the node.

5.5 Storage

The default log files are stored in folders under *%ProgramData%\Qlik\Sense\Log*. The local log configuration file can be used to set up the logging so that the log files are also stored in another location.

Log folder

The following table describes the contents of the *%ProgramData%\Qlik\Sense\Log* folder.

Folder	**Sub-folder**	**Files**	**Description**
\AppMigration			This folder contains log files related to the Migration Service.
\BrokerService			This folder contains log files related to the Broker Service.
\DataProfiling			This folder contains log files related to the Data Profiling Service.
\Engine		*<MachineName>_Exit_Engine_<Date>.txt*	This is a temporary log file that is used by the the Qlik Sense Engine Service (QES) until the log pipe to the Qlik Sense Repository Service (QRS) is up and running. This log file is not archived.

Folder	Sub-folder	Files	Description
		<MachineName>_Start_Engine_<Date>.txt	This is a temporary log file that is used by the the Qlik Sense Engine Service (QES) until the log pipe to the Qlik Sense Repository Service (QRS) is up and running. This log file is not archived.
	\Audit	*<MachineName>_AuditActivity_<Service>.txt*	This log tracks user-related actions.
		<MachineName>_AuditSecurity_<Service>.txt	This log contains information on security-related actions.
	\System	*<MachineName>_Service_<Service>.txt*	This log contains information on service and system operations, including all errors.
	\Trace	*<MachineName>_<Facility>_<Service>.txt*	The trace log files are stored in this folder. See: *Trace logs (page 137)*
\HubService			This folder contains log files related to the Hub Service.
\Printing	*\Audit*	*<MachineName>_AuditActivity_<Service>.txt*	This log tracks user-related actions.
		<MachineName>_AuditSecurity_<Service>.txt	This log contains information on security-related actions.
	\System	*<MachineName>_Service_<Service>.txt*	This log contains information on service and system operations, including all errors.
	\Trace	*<MachineName>_<Facility>_<Service>.txt*	The trace log files are stored in this folder. See: *Trace logs (page 137)*
\Proxy	*\Audit*	*<MachineName>_AuditActivity_<Service>.txt*	This log tracks user-related actions.

Folder	Sub-folder	Files	Description
		<MachineName>_ AuditSecurity_ <Service>.txt	This log contains information on security-related actions.
	\System	*<MachineName>_ Service_ <Service>.txt*	This log contains information on service and system operations, including all errors.
	\Trace	*<MachineName>_ <Facility>_ <Service>.txt*	The trace log files are stored in this folder. See: *Trace logs (page 137)*
\Repository	*\Audit*	*<MachineName>_ AuditActivity_ <Service>.txt*	This log tracks user-related actions.
		<MachineName>_ AuditSecurity_ <Service>.txt	This log contains information on security-related actions.
	\System	*<MachineName>_ Service_ <Service>.txt*	This log contains information on service and system operations, including all errors.
	\Trace	*<MachineName>_ <Facility>_ <Service>.txt*	The trace log files are stored in this folder. See: *Trace logs (page 137)*
\Scheduler	*\Audit*	*<MachineName>_ AuditActivity_ <Service>.txt*	This log tracks user-related actions.
		<MachineName>_ AuditSecurity_ <Service>.txt	This log contains information on security-related actions.
	\System	*<MachineName>_ Service_ <Service>.txt*	This log contains information on service and system operations, including all errors.
	\Trace	*<MachineName>_ <Facility>_ <Service>.txt*	The trace log files are stored in this folder. See: *Trace logs (page 137)*
\Script			This folder contains log files related to app reloads.
\WebExtensionService			This folder contains log files related to the Web Extension Service.

Archived log files

Archived log files are by default stored in *%ProgramData%\Qlik\Sense\Repository\Archived Logs* on the central node in the Qlik Sense site. Archived log files have the file extension *.log*, whereas active log files have the extension *.txt*.

See also:

🗋 *Local log configuration file (page 153)*

5.6 Naming

The Qlik Sense log files are named in accordance to the following file rollover procedure:

1. The log is stored in a file named *<MachineName>_<LogType>_<Service>.txt*.
2. When the file is full or a pre-defined amount of time has passed, the file extension is automatically changed to *.log* and a time stamp is appended to the file name for uniqueness and archiving. This means that the new file name becomes *<MachineName>_<LogType>_<Service>_<YYYY-MM-DDTHH.mm.ss>Z.log*. The file is then moved to the repository database on the central node by the Qlik Sense Repository Service (QRS) and archived.
3. A new log file, named *<MachineName>_<LogType>_<Service>.txt*, is created.

> ⚠ *If the .log file is deleted before being copied to the repository database on the central node, the file is lost and cannot be recreated.*

The format of the file name is as follows:

- *<MachineName>* = Name of the server where the log was created.
- *<LogType>* = The type of events that are covered by the log.
- *<Service>* = The service that the log originates from (for example, Proxy or Repository).
- *<YYYY-MM-DDTHH.mm.ss>Z* = Time stamp for when the log file was closed for new entries, where:
  - *YYYY*: Year
  - *MM*: Month
  - *DD*: Day in month
  - *T*: Delimiter, time designator
  - *HH*: Hour
  - *mm*: Minutes
  - *ss*: Seconds
  - *Z*: UTC designator, indicates that the time stamp is in UTC format

5.7 Rows

The first row of each log file contains a header that, in turn, contains the names of all fields, separated by tabs.

Each log entry is one row and the characters listed in the following table are replaced with Unicode characters.

Character	Unicode replacement	Description
\t	\u21d4	Symbol for horizontal tabulation, HT.
\n	\u2193	Symbol for line feed, LF.
\f	\u2192	Symbol for form feed, FF.
\r	\u21b5	Symbol for carriage return, CR.

5.8 Fields

This section describes the fields in the Qlik Sense log files.

Audit activity log

The following table lists the fields in the audit activity log, *<MachineName>_AuditActivity_<Service>.txt*.

The Audit activity log does not include a Severity field. This is because all rows in the log have the same log level.

Field	Format	Description
Sequence#	Int	1 - 2147483647 by default, but can be configured using custom logging as described in *Appenders (page 150)*. Each row in the log starts with a sequence number that is used to ensure that the log is not tampered with (that is, that no rows are inserted or deleted). The sequence number wraps a) when the last sequence number is reached, or b) when the logging, for some reason, is restarted without the last sequence number being reached.
ProductVersion	String	The version number of the Qlik Sense service (for example, 1.2.1.3).

Field	Format	Description
Timestamp	ISO 8601	Timestamp in ISO 8601 format, `YYYYMMDDThhmmss.fffK`, where: • `YYYY`: Year • `MM`: Month • `DD`: Day in month • `T`: Delimiter • `hh`: Hour • `mm`: Minutes • `ss`: Seconds • `fff`: Milliseconds • `K`: Time zone offset For example, 20110805T145657.000+0200 means year 2011, month 8, day 5 at 14:56:57 GMT+2.
Hostname	String	The name of the server.
Id	String	A unique identifier of the log entry (added by Log4net).
Description	String	A human-readable message that summarizes the action in the system. Format: Command=<CommandName>;Result=<ReturnCode (Int)>;ResultText=<Description, Success, or Error message>
ProxySessionId	String	The ID of the proxy session. 0 = Internal system command or a command that does not go through the QPS
ProxyPackageId	String	A unique ID of each HTTP(S) package that passes through the Qlik Sense Proxy Service (QPS). 0 = Internal system command or a command that does not go through the QPS

Field	Format	Description
RequestSequenceId	String	The combination of RequestSequenceId and ProxyPackageId is unique for every row in a log file and creates the timeline for the proxy session. The combination also forms a primary key in the log file. The initial RequestSequenceId is an integer. Subrequests are linked to the initial request by adding a dot and an ID for the subrequest: • Initial request: RequestSequenceId = 1 • Subrequest 1 based on the initial request: RequestSequenceId = 1.0 • Subrequest 2 based on the initial request: RequestSequenceId = 1.1 0 = Internal system command or a command that does not go through the Qlik Sense Engine Service (QES)
UserDirectory	String	The user directory linked to the logged in Qlik Sense user.
UserId	String	The Qlik Sense user that initiated the command. System = Internal system command
ObjectId	String	The internal ID of the object. Used to link system actions to user actions. 0 = Cannot get the ID of the object In some cases the ObjectId field contains multiple IDs, separated by the "|" (pipe) sign. **Example: ObjectId field containing multiple IDs** Log event: Start reload task Contents of the ObjectId field: ed5715cd-2d7f-44ec-825f-44084efb3443|d63c7e4e-6089-4314-b60f-ed47ba6c35cc • First ID: The ID of the task. • Second ID: The ID of the app.

Field	Format	Description
ObjectName	String	The human-readable name of the object. The ObjectName is linked to the ObjectId. Not available = Cannot link the ObjectName to the ObjectId or the ObjectId is missing In some cases the ObjectName field contains multiple names. **Example: ObjectName field containing multiple names** Log event: Start reload task Contents of the ObjectName field: MyReload|MyApp • First identifier (MyReload): The name of the task. • Second identifier (MyApp): The name of the app. The list of ObjectNames always matches the list of ObjectIds, meaning that the ObjectName in the first position is identified by the ID in the corresponding position of the ObjectId field. In this example the following IDs apply (see also the description of the ObjectId field): • MyReload = ed5715cd-2d7f-44ec-825f-44084efb3443 • MyApp = d63c7e4e-6089-4314-b60f-ed47ba6c35cc
Service	String	The Qlik Sense service on the server that hosts the process.
Origin	String	The origin of the request: • AppAccess • ManagementAccess • Not available
Context	String	The context of the command. The context can be a URL that is linked to the command or a short version of the module path linked to the command.
Command	String	The core name of the use case or system command.
Result	String	Return code: • 0, 200 - 226: Success • Any other number: Error
Message	String	Text that describes the log entry. If the request is successful, this field contains "success".
Id2	String	A unique row identifier (the checksum is added by Log4Net).

Audit security log

The following table lists the fields in the audit security log, *<MachineName>_AuditSecurity_<Service>.txt*.

This log is not available for the Qlik Sense Engine Service (QES).

The Audit security log does not include a Severity field. This is because all rows in the log have the same log level.

Field	Format	Description
Sequence#	Int	1 - 2147483647 by default, but can be configured using custom logging as described in *Appenders (page 150)*. Each row in the log starts with a sequence number that is used to ensure that the log is not tampered with (that is, that no rows are inserted or deleted). The sequence number wraps a) when the last sequence number is reached, or b) when the logging, for some reason, is restarted without the last sequence number being reached.
ProductVersion	String	The version number of the Qlik Sense service (for example, 1.2.1.3).
Timestamp	ISO 8601	Timestamp in ISO 8601 format, `YYYYMMDDThhmmss.fffK`, where: • `YYYY`: Year • `MM`: Month • `DD`: Day in month • `T`: Delimiter • `hh`: Hour • `mm`: Minutes • `ss`: Seconds • `fff`: Milliseconds • `K`: Time zone offset For example, 20110805T145657.000+0200 means year 2011, month 8, day 5 at 14:56:57 GMT+2.
HostName	String	The name of the server.
Id	GUID	A unique identifier of the log entry (added by Log4net).
Description	String	A human-readable message that summarizes the action in the system. Format: Command=<CommandName>;Result=<ReturnCode (Int)>;ResultText=<Description, Success, or Error message>

Field	Format	Description
ProxySessionId	String	The ID of the proxy session. 0 = Internal system command or a command that does not go through the QPS
ProxyPackageId	String	A unique ID of each HTTP(S) package that passes through the Qlik Sense Proxy Service (QPS). 0 = Internal system command or a command that does not go through the QPS
RequestSequenceId	String	The combination of RequestSequenceId and ProxyPackageId is unique for every row in a log file and creates the timeline for the proxy session. The combination also forms a primary key in the log file. The initial RequestSequenceId is an integer. Subrequests are linked to the initial request by adding a dot and an ID for the subrequest: • Initial request: RequestSequenceId = 1 • Subrequest 1 based on the initial request: RequestSequenceId = 1.0 • Subrequest 2 based on the initial request: RequestSequenceId = 1.1 0 = Internal system command or a command that does not go through the Qlik Sense Engine Service (QES)
UserDirectory	String	The user directory linked to the logged in Qlik Sense user. System = Internal system command
UserId	String	The Qlik Sense user that initiated the command. System = Internal system command

Field	Format	Description
ObjectId	String	The internal ID of the object. Used to link system actions to user actions. 0 = Cannot get the ID of the object In some cases the ObjectId field contains multiple IDs, separated by the "|" (pipe) sign. **Example: ObjectId field containing multiple IDs** Log event: Start reload task Contents of the ObjectId field: ed5715cd-2d7f-44ec-825f-44084efb3443|d63c7e4e-6089-4314-b60f-ed47ba6c35cc • First ID: The ID of the task. • Second ID: The ID of the app.
ObjectName	String	The human-readable name of the object. The ObjectName is linked to the ObjectId. Not available = Cannot link the ObjectName to the ObjectId or the ObjectId is missing In some cases the ObjectName field contains multiple names. **Example: ObjectName field containing multiple names** Log event: Start reload task Contents of the ObjectName field: MyReload|MyApp • First identifier (MyReload): The name of the task. • Second identifier (MyApp): The name of the app. The list of ObjectNames always matches the list of ObjectIds, meaning that the ObjectName in the first position is identified by the ID in the corresponding position of the ObjectId field. In this example the following IDs apply (see also the description of the ObjectId field): • MyReload = ed5715cd-2d7f-44ec-825f-44084efb3443 • MyApp = d63c7e4e-6089-4314-b60f-ed47ba6c35cc
SecurityClass	String	A categorization of the security-related information: • Security: Access to resources, authentication, authorization • License: License access, license usage, license allocation • Certificate: Certificate-related information
ClientHostAddress	String	The hostname/IP address of the client.

Field	Format	Description
Service	String	The Qlik Sense service on the server that hosts the process.
Origin	String	The origin of the request: • AppAccess • ManagementAccess • Not available
Context	String	The context of the command. The context can be a URL that is linked to the command or a short version of the module path linked to the command.
Command	String	The core name of the use case or system command.
Result	String	Return code: • 0, 200 - 226: Success • Any other number: Error
Message	String	Text that describes the log entry. If the request is successful, this field contains "success".
Checksum	ID	Each row has a checksum. The security log file also includes a file signature.

Service log

The following table lists the fields in the service log, *<MachineName>_Service_<Service>.txt*.

Field	Format	Description
Sequence#	Int	1 - 2147483647 by default, but can be configured using custom logging as described in *Appenders (page 150)*. Each row in the log starts with a sequence number that is used to ensure that the log is not tampered with (that is, that no rows are inserted or deleted). The sequence number wraps a) when the last sequence number is reached, or b) when the logging, for some reason, is restarted without the last sequence number being reached.
ProductVersion	String	The version number of the Qlik Sense service (for example, 1.2.1.3).

Field	Format	Description
Timestamp	ISO 8601	Timestamp in ISO 8601 format, `YYYYMMDDThhmmss.fffK`, where: • `YYYY`: Year • `MM`: Month • `DD`: Day in month • `T`: Delimiter • `hh`: Hour • `mm`: Minutes • `ss`: Seconds • `fff`: Milliseconds • `K`: Time zone offset For example, 20110805T145657.000+0200 means year 2011, month 8, day 5 at 14:56:57 GMT+2.
Severity	String	Row log level, can be configured using custom logging as described in *Appenders (page 150)*: • Debug: Information useful to developers for debugging purposes. This level is not useful during normal operation as it generates vast amounts of logging information. • Info: Normal operational messages that may be harvested for reporting, measuring throughput, and so on. No action is required. • Warn: Not an error message, but an indication that an error will occur, if no action is taken (for example, the file system is 85% full). • Error: Messages regarding unexpected situations and errors that prevent the server from operating normally. • Fatal: Messages that the Qlik Sense service or application has to shut down in order to prevent data loss.
HostName	String	The hostname of the server that runs the process or executes the task.
Id	GUID	A unique identifier of the log entry (added by Log4net).
Description	String	A human-readable message that summarizes the action in the system. Format: Command=<CommandName>;Result=<ReturnCode (Int)>;ResultText=<Description, Success, or Error message>

Field	Format	Description
ProxySessionId	String	The ID of the proxy session. 0 = Internal system command or a command that does not go through the QPS
ProxyPackageId	String	A unique ID of each HTTP(S) package that passes through the Qlik Sense Proxy Service (QPS). 0 = Internal system command or a command that does not go through the QPS
RequestSequenceId	String	The combination of RequestSequenceId and ProxyPackageId is unique for every row in a log file and creates the timeline for the proxy session. The combination also forms a primary key in the log file. The initial RequestSequenceId is an integer. Subrequests are linked to the initial request by adding a dot and an ID for the subrequest: • Initial request: RequestSequenceId = 1 • Subrequest 1 based on the initial request: RequestSequenceId = 1.0 • Subrequest 2 based on the initial request: RequestSequenceId = 1.1 0 = Internal system command or a command that does not go through the Qlik Sense Engine Service (QES)
UserDirectory	String	The user directory linked to the logged in Qlik Sense user. System = Internal system command
UserId	String	The Qlik Sense user that initiated the command. System = Internal system command

Field	Format	Description
ObjectId	String	The internal ID of the object. Used to link system actions to user actions. 0 = Cannot get the ID of the object In some cases the ObjectId field contains multiple IDs, separated by the "|" (pipe) sign. **Example: ObjectId field containing multiple IDs** Log event: Start reload task Contents of the ObjectId field: ed5715cd-2d7f-44ec-825f-44084efb3443|d63c7e4e-6089-4314-b60f-ed47ba6c35cc • First ID: The ID of the task. • Second ID: The ID of the app.
ObjectName	String	The human-readable name of the object. The ObjectName is linked to the ObjectId. Not available = Cannot link the ObjectName to the ObjectId or the ObjectId is missing In some cases the ObjectName field contains multiple names. **Example: ObjectName field containing multiple names** Log event: Start reload task Contents of the ObjectName field: MyReload|MyApp • First identifier (MyReload): The name of the task. • Second identifier (MyApp): The name of the app. The list of ObjectNames always matches the list of ObjectIds, meaning that the ObjectName in the first position is identified by the ID in the corresponding position of the ObjectId field. In this example the following IDs apply (see also the description of the ObjectId field): • MyReload = ed5715cd-2d7f-44ec-825f-44084efb3443 • MyApp = d63c7e4e-6089-4314-b60f-ed47ba6c35cc
Service	String	The Qlik Sense service on the server that hosts the process.
Origin	String	The origin of the request: • AppAccess • ManagementAccess • Not available

Field	Format	Description
Context	String	The context of the command. The context can be Internal System command or User Activity command (based on URL for the command).
Command	String	The core name of the use case or system command.
Result	Int	Return code: • 0, 200 - 226: Success • Any other number: Error
Message	String	Text that describes the log entry. If the request is successful, this field contains "success".
Id2	String	A unique row identifier (the checksum is added by Log4Net).

Qlik Sense Engine Service log fields

The following table lists the fields that are unique for the Qlik Sense Engine Service (QES) logs.

Field	Format	Description
EngineTimestamp	ISO 8601	The date and time when the QES wrote the log message to file. Timestamp in ISO 8601 format, `YYYYMMDDThhmmss.fffK`, where: • `YYYY`: Year • `MM`: Month • `DD`: Day in month • `T`: Delimiter • `hh`: Hour • `mm`: Minutes • `ss`: Seconds • `fff`: Milliseconds • `K`: Time zone offset For example, 20110805T145657.000+0200 means year 2011, month 8, day 5 at 14:56:57 GMT+2.
EngineVersion	String	The version number of the QES that executed the request.

5.9 Trace logs

The legacy logging framework is still available in Qlik Sense, but the logs are as of Qlik Sense version 2.0 referred to as trace logs. The log files remain the same, but they are stored in a new location.

Storage

The trace log files are stored in the *%ProgramData%\Qlik\Sense\Log\<Service>\Trace* folder.

Naming

The trace log files are named in accordance to the following file rollover procedure:

1. The log is stored in a file named *<MachineName>_<Facility>_<Service>.txt*.
2. When the file is full or a pre-defined amount of time has passed, the file extension is automatically changed to *.log* and a time stamp is appended to the file name for uniqueness and archiving. This means that the new file name becomes *<MachineName>_<Facility>_<Service>_<YYYY-MM-DDTHH.mm.ss>Z.log*. The file is then moved to the repository database on the central node by the Qlik Sense Repository Service (QRS) and archived.
3. A new log file, named *<MachineName>_<Facility>_<Service>.txt*, is created.

If the .log file is deleted before being copied to the repository database on the central node, the file is lost and cannot be recreated.

The format of the file name is as follows:

- *<Machine>* = Name of the server where the log was created.
- *<Facility>* = The type of events that are covered by the log.

  See: *Logger (page 141)*
- *<Service>* = The service that the log originates from (for example, Proxy or Repository).
- *<YYYY-MM-DDTHH.mm.ss>Z* = Time stamp for when the log file was closed for new entries, where:
  - *YYYY*: Year
  - *MM*: Month
  - *DD*: Day in month
  - *T*: Delimiter, time designator
  - *HH*: Hour
  - *mm*: Minutes
  - *ss*: Seconds
  - *Z*: UTC designator, indicates that the time stamp is in UTC format

See also:

Logger (page 141)

Rows

The first row of each log file contains a header that, in turn, contains the names of all fields, separated by tabs.

Each log entry is one row and the characters listed in the following table are replaced with Unicode characters.

Character	Unicode replacement	Description
\t	\u21d4	Symbol for horizontal tabulation, HT.
\n	\u2193	Symbol for line feed, LF.
\f	\u2192	Symbol for form feed, FF.
\r	\u21b5	Symbol for carriage return, CR.

Fields

This section describes the fields in the trace log files.

Common fields

The following table lists the fields (in order of appearance) included in all trace log files.

Field	Description
Sequence#	1 - 2147483647 by default, but can be configured using custom logging as described in *Appenders (page 150)*. Each row in the log starts with a sequence number that is used to ensure that the log is not tampered with (that is, that no rows are inserted or deleted). The sequence number wraps either when the last sequence number is reached or when the logging, for some reason, is restarted without the last sequence number being reached.
Timestamp	Timestamp in ISO 8601 format, `YYYYMMDDThhmmss.fffK`, where: • `YYYY`: Year • `MM`: Month • `DD`: Day in month • `T`: Delimiter • `hh`: Hour • `mm`: Minutes • `ss`: Seconds • `fff`: Milliseconds • `K`: Time zone offset For example, 20110805T145657.000+0200 means year 2011, month 8, day 5 at 14:56:57 GMT+2.

Field	Description
Level	Row log level, can be configured using custom logging as described in *Appenders (page 150)*: - Debug: Information useful to developers for debugging purposes. This level is not useful during normal operation since it generates vast amounts of logging information. - Info: Normal operational messages that may be harvested for reporting, measuring throughput, and so on. No action required. - Warn: Not an error message, but an indication that an error may occur, if no action is taken (for example, the file system is 85% full). Each item must be resolved within a given time. - Error: Non-urgent failures that are relayed to developers or administrators. Each item must be resolved within a given time. - Fatal: Indicates a failure in a primary system (for example, loss of primary ISP connection) and must be corrected immediately. - Off: No logs, except for license logs, are produced.
Hostname	Server name.

Field	Description
Logger	Logger in `<Facility>.<Service>.<Fully qualified name of class>` format, where: - `<Facility>`: - Application: Log events that are related to the app running in Qlik Sense. - Audit: Log events that provide an audit trail of a user's activities and administration of the Qlik Sense platform. - Exit: Log events that are related to the shutdown of the Qlik Sense Engine Service (QES). - License: Log events that are related to the Qlik Sense license. - ManagementConsole: Log events that are related to the Qlik Management Console (QMC). - Performance: Log events that are related to the performance of the Qlik Sense platform or app. - QixPerformance: Log events that are related to the performance of the QIX protocol in the QES. - Security: Log events that are related to security issues. - Session: Log events that are related to the termination of a proxy session. - Synchronization: Log events that are related to the synchronization of the Qlik Sense Repository Service (QRS) instances in a multi-node site. - System: Log events that are related to the Qlik Sense platform and not to the app running on the platform (for example, log messages related to the QMC, QRS, Qlik Sense Proxy Service (QPS), and so on). - TaskExecution: Log events that are related to the execution of tasks by the Qlik Sense Scheduler Service (QSS). - Traffic: Log events that are related to debugging. - UserManagement: Log events that are related to the management of the users. - `<Service>`: The Qlik Sense service that the log originates from (for example, QRS or QPS). - `<Fully qualified name of class>`: Indicates the part of the service that generated the log message.
Thread	Thread name or Managed Thread ID (if available).
Id	Globally Unique Identifier (GUID) for the log message.
ServiceUser	Name of the user or account used by the Qlik Sense service.
Message	Log message.

Field	Description
Exception	Exception message. *This field is only present when there is an exception message.*
StackTrace	A trace to the place in Qlik Sense where the exception occurred. *This field is only present when the Exception field is present.*
ProxySessionId	The ID of the proxy session for the user. *This field is not present in all log files.*
Id2 or Checksum	The last field in a log entry either contains an Id2 or a Checksum: • Id2: Log message GUID (same as Id described earlier). This is the normal value. • Checksum: To protect logs that contain sensitive information (for example, audit, security, and license logs) from tampering, the last field in such log entries contains a cryptographic hash of the entire row up to the hash itself.

See also:

Synchronization (page 16)

Additional fields

The common fields are present in all trace log files. Some trace logs contain additional fields, which are listed in this section. In addition, optional fields can be defined.

Application log

Qlik Sense Repository Service (QRS)

The following fields are specific to the Application log for the QRS:

- Application: The name of the application (if there is a name to associate with the log entry).

Qlik Sense Scheduler Service (QSS)

The following fields are specific to the Application log for the QSS:

- Application: The name of the application (if there is a name to associate with the log entry).

See also:

Common fields (page 139)

Audit log

Qlik Sense Repository Service (QRS)

The following fields are specific to the Audit log for the QRS:

- Action: The action that the user performed (add, update, delete, export).
- ActiveUserDirectory: The user directory for the user.
- ActiveUserId: The ID of the user.
- ResourceId: The ID of the resource on which the user performed the action.

Qlik Sense Proxy Service (QPS)

The following fields are specific to the Audit log for the QPS:

- ConnectionId: The ID of the connection.

  See: ActiveConnections field in *Performance log (page 144)*
- ActiveUserDirectory: The user directory for the user.
- ActiveUserId: The ID of the user.
- TicketId: The ID of the login ticket that was issued for the user. The ticket ID exists until it is consumed by the QPS.
- IpAddress: The IP address of the client.
- AppId: The ID of the app (empty if no app is loaded).
- TargetHost: The call from the client is forwarded to a Qlik Sense Engine Service (QES) or QRS. This field contains the name of the machine on which the service is running.
- VirtualProxy: The virtual proxy prefix in `{prefix}` format.

Qlik Sense Engine Service (QES)

The following fields are specific to the Audit log for the QES:

- ActiveUserDirectory: The user directory for the user.
- ActiveUserId: The ID of the user.
- EngineTimestamp: The time when the QES wrote the log message to file.
- EngineThread: The ID of the thread that was used when the QES wrote the log message to file.
- ProcessId: The ID of the QES process from which the log message originates.
- ServerStatus: The time when the QES started.
- AppId: The ID of the app.
- Type: The type of operation that the user performed to generate the audit message.
- Qlik Sense User: The user who generated the audit message.

See also:

Common fields (page 139)

License log

Qlik Sense Repository Service (QRS)

The following fields are specific to the License log for the QRS:

- AccessTypeId: The ID of the access type entity.
- AccessType: The name of the access type (LoginAccess or UserAccess).
- Operation: The operation that was performed (Add, Update, Delete, UsageGranted, UsageDenied, Available, Timedout, or Unquarantined).
- UserName: The name of the user (who, for example, uses an access pass).
- UserId: The ID of the user in Qlik Sense.

See also:

Common fields (page 139)

Performance log

Qlik Sense Repository Service (QRS)

The following fields are specific to the Performance log for the QRS:

- Tracenumber: A unique ID for the call to the QRS REST API.
- Httpmethod: The HTTP method that was used (Get, Put, Post, or Delete).
- Url: The URL that was used.
- Resourcetype: The type of resource.
- Method: The backend code that was called.
- Elapsedmilliseconds: The time (in milliseconds) to complete the call to the QRS REST API.

Example: Get http://mytest/cars/4

- Httpmethod: Get
- Url: http://mytest/cars/4
- Resourcetype: cars
- Method: get/cars/{0}

Qlik Sense Proxy Service (QPS)

The following fields are specific to the Performance log for the QPS:

- ActiveConnections: The number of active connections (in any form or shape) from the client.

  A connection is a stream (that is, a socket) between a Qlik Sense client and the Qlik Sense Proxy Service (QPS). This stream is often connected to another stream, which runs from the QPS to the Qlik Sense Repository Service (QRS) or the Qlik Sense Engine Service (QES). The two streams allow the client to communicate with the QRS or the QES.

- ActiveStreams: The number of active data streams (that is, sockets), either from the browser to the QPS or from the QPS to the QRS or the QES.
- ActiveSessions: The number of active sessions in the QPS.

  A Qlik Sense user gets a proxy session when the user has been authenticated. The session is terminated after a certain period of inactivity.
- LoadBalancingDecisions: The number of users who currently have at least one engine session.
- PrintingLoadBalancingDecisions: The number of users who have been load balanced to the Qlik Sense Printing Service (QPR).
- Tickets: The number of issued login tickets that have not yet been consumed.
- ActiveClientWebsockets: The number of active WebSockets between the client and the QPS.
- ActiveEngineWebsockets: The number of active WebSockets between the QPS and the target Qlik Sense service.

The logging entries are also available as metrics; see Qlik Sense Proxy Service (page 22).

Qlik Sense Engine Service (QES)

Each entry (that is, row) in the Performance log corresponds to a snapshot (that is, a number of measurements) of the performance of the QES at the given point in time.

The following fields are specific to the Performance log for the QES:

- ActiveUserDirectory: The user directory for the user.
- ActiveUserId: The ID of the user.
- EngineTimestamp: The time when the QES wrote the log message to file.
- EngineThread: The ID of the thread that was used when the QES wrote the log message to file.
- ProcessId: The ID of the QES process from which the log message originates.
- Exe Type: The configuration type (release or debug version) of the QES process.
- Exe Version: The version number of the QES process.
- Server Started: The time when the QES started.
- Entry Type: The reason (Server Starting, Normal, or Server Shutting Down) for the log entry in the Performance log.
- ActiveDocSessions: The number of active engine sessions at the given point in time.
- DocSessions: The number of engine sessions at the given point in time.
- ActiveAnonymousDocSessions: The number of active anonymous engine sessions at the given point in time.
- AnonymousDocSessions: The number of anonymous engine sessions at the given point in time.
- ActiveTunneledDocSessions: The number of active tunneled engine sessions at the given point in time.
- TunneledDocSessions: The number of tunneled engine sessions at the given point in time.
- DocSessionStarts: The number of started engine sessions since the previous snapshot.
- ActiveDocs: The number of active apps in the QES at the given point in time.

- RefDocs: The number of apps in the QES at the given point in time.
- LoadedDocs: The number of loaded apps in the QES at the given point in time.
- DocLoads: The number of app loads in the QES since the previous snapshot.
- DocLoadFails: The number of failed app loads in the QES since the previous snapshot.
- Calls: The number of calls to the QES since the previous snapshot.
- Selections: The number of selections in the QES since the previous snapshot.
- ActiveIpAddrs: The number of IP addresses of active connected clients in the QES at the given point in time.
- IpAddrs: The number of IP addresses of all connected clients in the QES at the given point in time.
- ActiveUsers: The number of active users in the QES at the given point in time.
- Users: The total number of users in the QES at the given point in time.
- CPULoad: A measurement of the load on the CPU on which the QES runs at the given point in time.
- VMCommitted(MB): The committed Virtual Memory (in megabytes) at the given point in time.
- VMAllocated(MB): The allocated Virtual Memory (in megabytes) at the given point in time.
- VMFree(MB): The freed Virtual Memory (in megabytes) at the given point in time.
- VMLargestFreeBlock(MB): The largest freed Virtual Memory block (in megabytes) at the given point in time.

See also:

Common fields (page 139)

QIX performance log

Qlik Sense Engine Service (QES)

The following fields are specific to the QIX performance log for the QES:

- ActiveUserDirectory: The user directory for the user.
- ActiveUserId: The ID of the user.
- EngineTimestamp: The time when the QES wrote the log message to file.
- EngineThread: The ID of the thread that was used when the QES wrote the log message to file.
- ProcessId: The ID of the QES process from which the log message originates.
- CServerId: The ID of the server instance that handled the request.
- SessionId: The ID of the engine session for which the QIX method call was made.
- Server Started: The time when the QES started.
- Method: The name of the QIX method that was called.
- RequestId: The ID of the request in which the QIX method call was handled.
- Target: The memory address of the target for the QIX method call.
- RequestException: The ID of an exception (if any) that occurred as a result of the QIX method call.
- ProcessTime: The amount of time that was needed to process the request.
- WorkTime: The amount of time that the request did actual work.
- LockTime: The amount of time that the request had to wait for an internal lock.

- ValidateTime: The amount of time that the request used for validation.
- Handle: The ID of the interface that handled the request. The interface can be Global, a certain sheet, a certain object, or similar.

See also:

🗋 *Common fields (page 139)*

Qlik Management Console log

The Qlik Management Console log is not created until there is an event (for example, an error message) for the Qlik Management Console (QMC) to write in the log.

Qlik Sense Repository Service (QRS)

The following fields are specific to the Qlik Management Console log for the QRS:

- Browser: The web browser that is used to run the QMC.

See also:

🗋 *Common fields (page 139)*

Session log

Qlik Sense Engine Service (QES)

The following fields are specific to the Session log for the QES:

- ActiveUserDirectory: The user directory for the user.
- ActiveUserId: The ID of the user.
- EngineTimestamp: The time when the QES wrote the log message to file.
- EngineThread: The ID of the thread that was used when the QES wrote the log message to file.
- ProcessId: The ID of the QES process from which the log message originates.
- Exe Type: The configuration type (release or debug version) of the QES process.
- Exe Version: The version number of the QES process.
- Server Started: The time when the QES started.
- AppId: The ID of the app that was loaded by the finished engine session.
- App Title: The title of the loaded app that was used during the finished engine session.
- Doc Timestamp: The last modified timestamp of the app that was loaded by the finished engine session.
- Qlik Sense User: The user that started the finished engine session.
- Exit Reason: The reason for the engine session to finish.
- Session Start: The time when the engine session started.
- Session Duration: The duration (in milliseconds) of the finished engine session.

- CPU Spent (s): The CPU time (in seconds) that was spent handling requests during the finished engine session.
- Bytes Received: The number of bytes of data that were received during the engine session.
- Bytes Sent: The number of bytes of data that were sent during the engine session.
- Calls: The number of calls that were made during the engine session.
- Selections: The number of selections that were made during the engine session.
- Authenticated User: The authenticated user that used the engine session.
- Client Machine Identification: The ID of the client machine that opened the engine session.
- Serial Number: The serial number that was used during the engine session.
- Client Type: The type of client that was used for the engine session.
- Client Build Version: The build version of the client.
- Secure Protocol: An on/off flag that indicates whether the protocol was run over a secure connection.

See also:

Common fields (page 139)

System log

Qlik Sense Scheduler Service (QSS)

The following fields are specific to the System log for the QSS:

- TaskName: The name of the task that was executed.
- TaskId: The ID of the task that was executed.
- User: The name of the user who executed the task. When the QSS starts a scheduled execution of a task, the QSS is listed as user.
- ExecutionId: A unique ID that identifies the execution of the task. A task gets a new ExecutionId every time it is executed.
- AppName: The name of the app that executed the task (if any).
- AppId: The ID of the app that executed the task (if any).

Qlik Sense Engine Service (QES)

The following fields are specific to the System log for the QES:

- ActiveUserDirectory: The user directory for the active user who was logged in when the log message was generated in the QES.
- ActiveUserId: The user ID for the active user who was logged in when the log message was generated in the QES.
- EngineTimestamp: The time when the QES wrote the log message to file.
- EngineThread: The ID of the thread that was used when the QES wrote the log message to file.
- ProcessId: The ID of the QES process from which the log message originates.
- Server Started: The time when the QES started.

See also:

🗋 *Common fields (page 139)*

Task execution log

Qlik Sense Scheduler Service (QSS)

The following fields are specific to the Task execution log for the QSS:

- TaskId: A unique ID of the task that was executed.
- TaskName: The name of the task that was executed.
- AppId: A unique ID of the app that executed the task (if any).
- AppName: The name of the app that executed the task (if any).
- ExecutionId: A unique ID that identifies the execution of a task. A task gets a new ExecutionId every time it is executed.
- ExecutionNodeId: A unique ID that identifies the node in the site on which the specific execution of the task was performed.
- Status: The result of the execution of the task (successful, failed, aborted, skipped, or retry).
- StartTime: The time when the execution of the task started.
- StopTime: The time when the execution of the task stopped.
- Duration: The time (in milliseconds) for the execution of the task to be completed.
- FailureReason: Empty, unless an error occurred during the execution of the task.

See also:

🗋 *Common fields (page 139)*

Traffic log

Qlik Sense Engine Service (QES)

The following fields are specific to the traffic log for the QES:

- ActiveUserDirectory: The user directory for the user.
- ActiveUserId: The ID of the user.
- EngineTimestamp: The time when the QES wrote the log message to file.
- EngineThread: The ID of the thread that was used when the QES wrote the log message to file.
- ProcessId: The ID of the QES process from which the log message originates.

See also:

🗋 *Common fields (page 139)*

5.10 Configuring the logging

The standard logging in Qlik Sense is configured using the Qlik Management Console (QMC).

Customized logging is setup using appenders and the local log configuration file, *LocalLogConfig.xml*.

Appenders

The logging in Qlik Sense implements a custom appender, QSRollingFileAppender, which is based on the log4net component. The custom appender is used internally by the Qlik Sense logging system.

QSRollingFileAppender and some of the built-in, predefined appenders in the log4net framework can be used to configure customized logging, which is specified in the local log configuration file, *LocalLogConfig.xml*.

QSRollingFileAppender can also log events in the local log file (for example, the Microsoft Windows event log) or send log information to a remote log server.

QSRollingFileAppender

QSRollingFileAppender inherits from `log4net.Appenders.FileAppender` and all parameters, except for `AppendToFile`, are also available to QSRollingFileAppender. QSRollingFileAppender stores the log files in accordance to the `MaxFileSize` and `MaxFileTime` parameters.

Configuring the appender

The QSRollingFileAppender configuration is as follows:

```
<appender name="MyQSRollingFileAppender"
type="Qlik.Sense.Logging.log4net.Appender.QSRollingFileAppender">
<param name="threshold" value="info" />
<param name="encoding" value="utf-8" />
<param name="file" value="C:/ProgramData/Qlik/Sense/Log/output.log"/>
<param name="maximumfiletime" value="720" />
<param name="maximumfilesize" value="512KB" />
<layout type="log4net.Layout.PatternLayout">
<converter>
  <param name="name" value="rownum" />
  <param name="type" value="Qlik.Sense.Logging.log4net.Layout.Pattern.CounterPatternConverter" />
</converter>
<converter>
  <param name="name" value="longIso8601date" />
  <param name="type"
value="Qlik.Sense.Logging.log4net.Layout.Pattern.Iso8601TimeOffsetPatternConverter" />
</converter>
<converter>
  <param name="name" value="hostname" />
  <param name="type" value="Qlik.Sense.Logging.log4net.Layout.Pattern.HostNamePatternConverter" />
</converter>
<converter>
  <param name="name" value="guid" />
  <param name="type" value="Qlik.Sense.Logging.log4net.Layout.Pattern.GuidPatternConverter" />
</converter>
```

```
<converter>
  <param name="name" value="user" />
  <param name="type"
value="Qlik.Sense.Logging.log4net.Layout.Pattern.ServiceUserNameCachedPatternConverter" />
</converter>
<converter>
  <param name="name" value="encodedmessage" />
  <param name="type" value="Qlik.Sense.Logging.log4net.Layout.Pattern.EncodedMessagePatternConverter"
/>
</converter>
<converter>
  <param name="name" value="encodedexception" />
  <param name="type"
value="Qlik.Sense.Logging.log4net.Layout.Pattern.EncodedExceptionPatternConverter" />
</converter>
<param name="ignoresexception" value="false" />
<param name="header"
value="Sequence#&#x9;Timestamp&#x9;Level&#x9;Hostname&#x9;Logger&#x9;Thread&#x9;Id&#x9;User&#x9;
Message&#x9;Exception&#x9;Id2&#xD;&#xA;" />
<param name="conversionpattern" value="%rownum
{9999}&#x9;%longIso8601date&#x9;%level&#x9;%hostname&#x9;%logger&#x9;%thread&#x9;
%guid&#x9;%user&#x9;%encodedmessage&#x9;%encodedexception{innermostmessage}&#x9;%guid%newline" />
</layout>
</appender>
```

Converters

`log4net.Layout.PatternLayout` and a couple of custom converters are used to format rows in logs based on QSRollingFileAppender:

- `Qlik.Sense.Logging.log4net.Layout.Pattern.CounterPatternConverter`: Add a sequence number to the log row. Parameter:
    - Integer: The last number of the sequence before it is reset.
- `Qlik.Sense.Logging.log4net.Layout.Pattern.Iso8601TimeOffsetPatternConverter`: Add a time stamp (local time with time offset in ISO 8601 format) to the log row.
- `Qlik.Sense.Logging.log4net.Layout.Pattern.HostNamePatternConverter`: Add the host name to the log row.
- `Qlik.Sense.Logging.log4net.Layout.Pattern.GuidPatternConverter`: Add a GUID to the log row.
- `Qlik.Sense.Logging.log4net.Layout.Pattern.ServiceUserNameCachedPatternConverter`: Add the username to the log row.
- `Qlik.Sense.Logging.log4net.Layout.Pattern.EncodedMessagePatternConverter`: Add the encoded message to the log row.
- `Qlik.Sense.Logging.log4net.Layout.Pattern.EncodedExceptionPatternConverter`: Add information on the logged exception to the log row. Parameter (one of the following):
    - MESSAGE: The message in the logged exception.
    - INNERMOSTMESSAGE: The message in the innermost exception of the logged exception.
    - SOURCE: The source of the exception (that is, the name of the app or the object that caused the error).
    - STACKTRACE: The stack trace for the exception.

- TARGETSITE: The target site for the exception (that is, the method that threw the current exception).
- HELPLINK: Link to the help file associated with the exception.

Built-in log4net appenders

In addition to the Qlik Sense custom appender, QSRollingFileAppender, the log4net framework comes with a set of built-in predefined appenders that also can be used in the local log configuration file, *LocalLogConfig.xml*:

- AdoNetAppender
- AnsiColorTerminalAppender
- AspNetTraceAppender
- ColoredConsoleAppender
- ConsoleAppender
- EventLogAppender
- FileAppender
- NetSendAppender
- RemoteSyslogAppender
- RemotingAppender
- RollingFileAppender
- SmtpAppender
- SmtpPickupDirAppender
- TelnetAppender
- UdpAppender

Each appender has its own set of parameters to control the output.

See also:

[Apache Logging Services]

Example: EventLogAppender

The following example shows how to use the EventLogAppender in the local log configuration file, *LocalLogConfig.xml*, for the Qlik Sense Proxy Service (QPS). In the example, all QPS audit log entries at warning level are sent to the Microsoft Windows event log.

```
<appender name="EventLogAppender" type="log4net.Appender.EventLogAppender" >
    <param name="threshold" value="warn" />
    <param name="applicationName" value="Qlik Sense Proxy Service" />
    <layout type="log4net.Layout.PatternLayout">
      <param name="conversionPattern" value="%message" />
    </layout>
  </appender>
     <logger name="Audit.Proxy">
                               <appender-ref ref="EventLogAppender" />
   </logger>
```

Example: SmtpAppender

The following example shows how to use the SmtpAppender in the local log configuration file, *LocalLogConfig.xml*, for the Qlik Sense Proxy Service (QPS). In the example, all QPS audit log entries at warning level are sent to an email address (`to@domain.com`).

```
<appender name="MyMailAppender" type="log4net.Appender.SmtpAppender">
<param name="threshold" value="warn" />
<param name="to" value="to@domain.com" />
<param name="from" value="from@domain.com" />
<param name="subject" value="test logging message" />
<param name="smtpHost" value="SMTPServer.domain.com" />
<param name="port" value="25" />
<param name="bufferSize" value="512" />
<param name="lossy" value="true" />
<layout type="log4net.Layout.PatternLayout">
<param name="conversionPattern" value="%newline%date %-5level %message%newline%newline%newline" />
</layout>
</appender>
     <logger name="Audit.Proxy">
                              <appender-ref ref="MyMailAppender" />
   </logger>
```

Local log configuration file

The logging in Qlik Sense can be set up to produce customized logging as an addition to the default logging.

To set up customized logging, create a local log configuration file named *LocalLogConfig.xml* in the *%ProgramData%\Qlik\Sense\<Service>* folder.

The logging defined by the local log configuration file does not affect the default logging.

Requirements

The requirements described in this section must be fulfilled for the customized logging to function properly.

Conforming to the XML schema

The local log configuration file must conform to the XML schema because the Qlik Sense Repository Service (QRS), Qlik Sense Proxy Service (QPS), and Qlik Sense Scheduler Service (QSS) have built-in schema validation.

If the local log configuration file is not accepted by the services, an error is logged in the System log.

Maximum file size

The size of the local log configuration file must not exceed 1 MB.

XML schema

The XML schema for the local log configuration file, *LocalLogConfig.xml*, is as follows:

In this example, the local log configuration file is configured to write the system logs at debug level in %ProgramData%\Qlik\Sense\Log\Proxy\Debug_System_Proxy.txt.

```
<?xml version="1.0"?>
<configuration>
   <appender name="LocalApp_AppenderSystem"
type="Qlik.Sense.Logging.log4net.Appender.QSRollingFileAppender">
      <param name="threshold" value="debug" />
      <param name="encoding" value="utf-8" />
      <param name="file" value="C:\ProgramData\Qlik\Sense\Log\Proxy\Debug_System_Proxy.txt" />
      <param name="maximumfiletime" value="720" />
      <param name="maximumfilesize" value="512KB" />
      <layout type="log4net.Layout.PatternLayout">
         <converter>
            <param name="name" value="rownum" />
            <param name="type"
value="Qlik.Sense.Logging.log4net.Layout.Pattern.CounterPatternConverter" />
         </converter>
         <converter>
            <param name="name" value="longIso8601date" />
            <param name="type"
value="Qlik.Sense.Logging.log4net.Layout.Pattern.Iso8601TimeOffsetPatternConverter" />
         </converter>
         <converter>
            <param name="name" value="hostname" />
            <param name="type"
value="Qlik.Sense.Logging.log4net.Layout.Pattern.HostNamePatternConverter" />
         </converter>
         <converter>
            <param name="name" value="guid" />
            <param name="type" value="Qlik.Sense.Logging.log4net.Layout.Pattern.GuidPatternConverter"
/>
         </converter>
         <converter>
            <param name="name" value="serviceuser" />
            <param name="type"
value="Qlik.Sense.Logging.log4net.Layout.Pattern.ServiceUserNameCachedPatternConverter" />
         </converter>
         <converter>
            <param name="name" value="encodedmessage" />
            <param name="type"
value="Qlik.Sense.Logging.log4net.Layout.Pattern.EncodedMessagePatternConverter" />
         </converter>
         <converter>
            <param name="name" value="encodedexception" />
            <param name="type"
value="Qlik.Sense.Logging.log4net.Layout.Pattern.EncodedExceptionPatternConverter" />
         </converter>
         <param name="ignoresexception" value="false" />
         <param name="header" value="Sequence#&#x9;Timestamp&#x9;Level&#x9;Hostname&#x9;
            Logger&#x9;Thread&#x9;Id&#x9;ServiceUser&#x9;Message&#x9;Exception&#x9;
            ActiveUserDirectory&#x9;ActiveUserId&#x9;ProxyTimestamp&#x9;ProxyThread&#x9;
            Id2&#xD;&#xA;" />
         <param name="conversionpattern" value="%rownum{9999}&#x9;%longIso8601date&#x9;
            %level&#x9;%hostname&#x9;%logger&#x9;%thread&#x9;%guid&#x9;%serviceuser&#x9;
            %encodedmessage{1000000}&#x9;%encodedexception{innermostmessage:1000000}&#x9;
            %property{ActiveUserDirectory}&#x9;%property{ActiveUserId}&#x9;
            %property{ProxyTimestamp}&#x9;%property{ProxyThread}&#x9;%guid%newline" />
      </layout>
   </appender>
   <logger name="System.Proxy">
      <appender-ref ref="LocalApp_AppenderSystem" />
   </logger>
</configuration>
```

See also:

🗋 *Converters (page 151)*

6 Licensing

The licensing in Qlik Sense is based on tokens, which are used to allocate access passes that allow users to access Qlik Sense. There are different types of access passes to choose from and each type corresponds to a specific consumption model for accessing Qlik Sense.

> *The tokens used in Qlik Sense are not compatible with the Client Access Licenses (CALs) used in QlikView. In addition, QlikView licenses cannot be used in Qlik Sense.*

6.1 License Enabler File

The Qlik Sense licensing is administered using a License Enabler File (LEF), which holds the number of tokens available for the central node in a site. This means that a Qlik Sense site needs at least one (1) LEF.

The LEF can be downloaded when the serial number and the control number have been entered in the Qlik Management Console (QMC). The LEF can also be pasted directly into the QMC, if, for example, no network connection is available.

Increase in tokens

When the number of tokens in the LEF increases (for example, when buying additional tokens), the new tokens are added to the pool of unallocated tokens that can be used to allocate access passes that allow users to access Qlik Sense.

Decrease in tokens

When the number of tokens in the LEF decreases, the following happens:

1. Unallocated tokens are removed.
2. If step 1 is not enough to meet the decreased number of tokens in the LEF, any tokens that are freed up by removal of access passes cannot be used for new allocations until the number of allocated tokens is below the new number set in the LEF.
   See: *Removing access passes (page 159)*

6.2 Access passes

The licensing in Qlik Sense is based on tokens, which are used to allocate access passes that allow users to access Qlik Sense. There are different types of access passes to choose from and each type corresponds to a specific consumption model for accessing Qlik Sense.

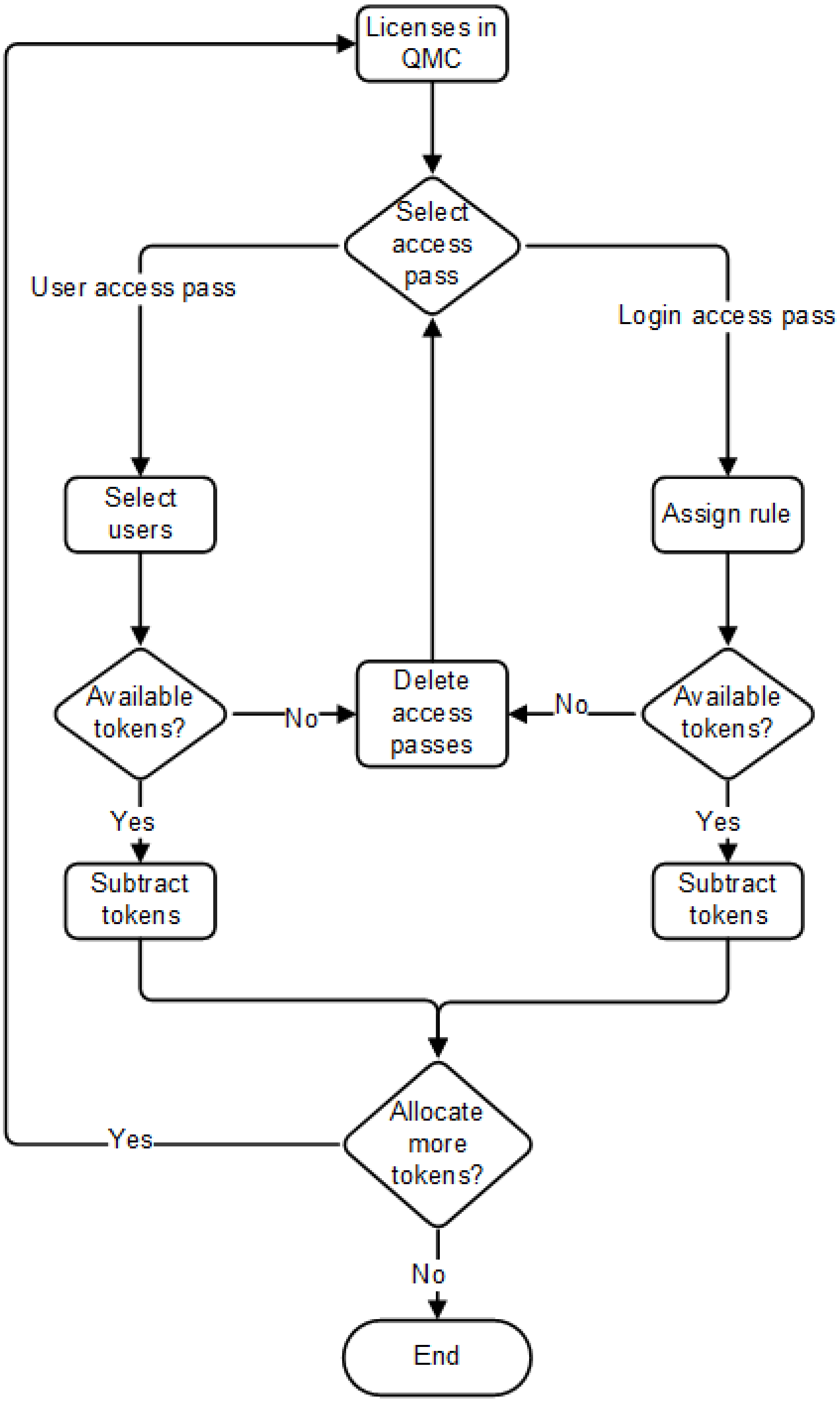
A user connection is the combination of device and browser that is used by a single user to connect to Qlik Sense. If a user who already has a user connection connects to Qlik Sense from another browser or device, an additional user connection is established.

The following table lists the types of access passes that are available in Qlik Sense.

Access type	Description
User access pass	This type of access pass allows a unique and identified user to access the hub. The access pass is valid within an entire Qlik Sense site. For example, if a user first connects to a node in the USA and then, at a later stage, connects to a node in the UK, the user consumes the same access pass, if the two nodes are connected to the same central node. See: *Architecture (page 12)* The maximum number of parallel user connections for a single user of this type of access pass is five (5). When a user with the maximum number of parallel user connections ends a connection (for example, by logging out) five minutes must pass before the user can use the access pass to add another connection (for example, by logging in). One (1) token corresponds to one (1) access pass. The access passes are allocated using the Qlik Management Console (QMC).
Login access pass	This type of access pass allows an identified or anonymous user to access the hub for a maximum of 60 continuous minutes per 28-day period. If the user exceeds the 60 minutes time limitation, the user connection does not time out. Instead, another login access pass is used. If no more login access passes are available, the user connection is discontinued. • If an identified user is disconnected, the user can re-connect and continue to use the same access pass, if re-connecting within the 60 minutes. • If an anonymous user is disconnected, the user gets a new access pass when re-connecting. The login access pass tracks the number of logins and runs over 28 days. For example, if 1000 logins are assigned to Group A, the users in Group A can use 1000 logins over 28 days. If 100 logins are consumed on Day 1, the 100 logins are available again on Day 29. The maximum number of parallel user connections for a single user of this type of access pass is five (5). Note that this only applies to identified users. An anonymous user can only have one (1) user connection. When a user with the maximum number of parallel user connections ends a connection (for example, by logging out) five minutes must pass before the user can use the access pass to add another connection (for example, by logging in). However, a user can have more connections than allowed by a single access pass by consuming additional access passes. One (1) token corresponds to ten (10) access passes. The access passes are allocated using login access groups in the QMC.

Allocation of access passes

The following figure shows how the Qlik Management Console (QMC) is used to manage the allocation of access passes.



Login and logout

Login

When a user logs in to Qlik Sense, an access pass of the applicable type is used to provide the user with access to Qlik Sense.

Logout

When a user logs out of Qlik Sense, the following happens depending on the type of access pass used:

- User access pass: The access pass is not affected when the user logs out.
- Login access pass: The access pass that was used to access Qlik Sense is considered to be used and will not be available for a new login until the period specified in *Login access pass (page 157)* has passed.

Removing access passes

This section describes how to free up tokens for new allocations of access passes by removing existing access passes in the Qlik Management Console (QMC).

User access pass

When a user access pass is removed, it enters a quarantine for seven (7) days, counting from the last time that the access pass was used. For example, if the access pass is used on January 10, the tokens used to allocate the access pass are not available for new allocations until January 18.

During the quarantine period, the original allocation of the access pass can be reinstated, which means that the quarantine period ends and the user can start using the access pass again.

Login access pass

When a login access group is removed, the tokens used to allocate the access pass become available in accordance to the following procedure:

1. For every ten (10) **unused** login access passes, one (1) token is freed up.
2. For every ten (10) login access passes that leave the **used** state after the period specified in *Login access pass (page 157)* has passed, one (1) token is freed up.

Disconnected node

A disconnected node is a rim node that fails to synchronize with the central node in a Qlik Sense site. A disconnected node continues to serve users to the best of its ability while waiting for a synchronization with the central node to take place.

Multi-deployment sites

This section describes how the Qlik Sense licensing is handled within multi-deployment sites, where apps are promoted from a development site to a test site and finally to a production site.

Development site

In a Qlik Sense deployment that includes a development site and a production site, two (2) License Enabler Files (LEF) are needed (that is, one per site).

Each node within the development site is licensed with one (1) access pass type (for example, user access passes), if only disconnected users are expected.

Test site

The LEF for a test site mirrors the LEF for a development site.

See also:

- *Deploying multi-node sites with synchronized persistence (page 45)*

Anonymous users

Anonymous users only use login access passes.

See also:

- *Login access pass (page 157)*

6.3 Licensing metrics

License metrics for the software are available at www.qlik.com/license-terms.