21CS Streams V7.1.0

Installation and Configuration Guide

Installation and Configuration Guide

Note

Before using this information and the product it supports, read the information in “Notices”.

Edition Notice

This edition applies to Version 7, Release 1, Modification 0 of 21CS Streams and to all subsequent releases and modifications until otherwise indicated in new editions.

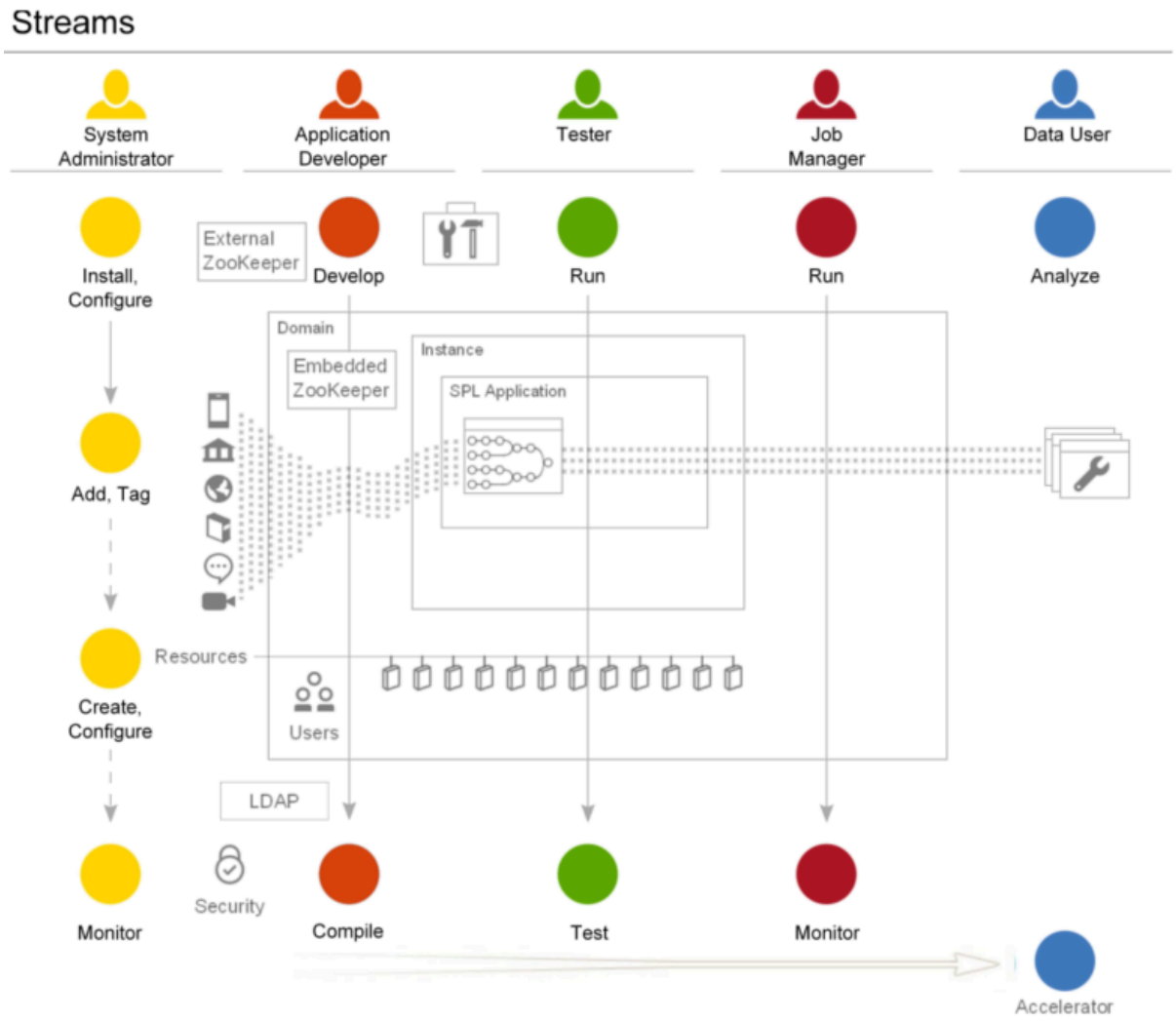
Part I. Overview

Part II. Installing Streams

Part I. Overview

Chapter 1. Introduction to Streams

21CS® Streams (referred to as Streams later on in this document) is a software platform that enables the development and execution of applications that process information in data streams. Streams enables continuous and fast analysis of massive volumes of moving data to help improve the speed of business insight and decision making.



1.1. Streams Features and Architecture

Streams consists of a programming language, an API, and an integrated development environment (IDE) for applications, and a runtime system that can run the applications on a single or distributed set of resources. The Streams Studio IDE includes tools for authoring and creating visual representations of stream processing applications.

Streams is designed to address the following data processing platform objectives:

- Parallel and high performance streams processing software platform that can scale over a range of hardware environments
- Automated deployment of stream processing applications on configured hardware

- Incremental deployment without restarting to extend stream processing applications
- Secure and auditable run time environment

The Streams architecture represents a significant change in computing system organization and capability. Streams provides a runtime platform, programming model, and tools for applications that are required to process continuous data streams. The need for such applications arises in environments where information from one to many data streams can be used to alert humans or other systems, or to populate knowledge bases for later queries.

Streams is especially powerful in environments where traditional batch or transactional systems might not be sufficient, for example:

- The environment must process many data streams at high rates.
- Complex processing of the data streams is required.
- Low latency is needed when processing the data streams.

Streams offers the *Streams Processing Language (SPL)* interface for users to operate on data streams. SPL provides a language and runtime framework to support stream processing applications. Users can create applications without needing to understand the lower-level stream-specific operations. SPL provides numerous operators, the ability to import data from outside Streams and export results outside the system, and a facility to extend the underlying system with user-defined operators. Many of the SPL built-in operators provide powerful relational functions such as Join and Aggregate.

Users can also develop stream processing applications in other supported languages, such as Java™ or Scala.

Deploying stream processing applications results in the creation of a dataflow graph, which runs across the distributed run time environment. As new workloads are submitted, Streams determines where to best deploy the operators to meet the resource requirements of both newly submitted and already running specifications. Streams continuously monitors the state and utilization of its computing resources. When stream processing applications are running, they can be dynamically monitored across a distributed collection of resources by using the Streams Console, Streams Studio, and `streamtool` commands.

Results from the running applications can be made available to applications that are running external to Streams by using Sink operators or edge adapters. For example, an application might use a TCPSink operator to send its results to an external application that visualizes the results on a map. Alternatively, it might alert an administrator to unusual or interesting events. Streams also provides many edge adapters that can connect to external data sources for consuming or storing data.

1.2. Streams Components

Streams consists of many components such as stream processing applications, domains, instances, and resources.

1.2.1. Streams Domains

A Streams domain is a logical grouping of resources in a network for common management and administration. To use Streams, you must create at least one domain.

There are two main types of domains: basic and enterprise. You can create both types of domains by using the `streamtool mkdomain` command.

A *basic domain* has a single Streams resource and user. This type of domain is typically used for test or development environments. Streams uses the Pluggable Authentication Module (PAM) authentication service for user authentication, and Apache ZooKeeper for managing and storing configuration information. By default, Streams uses an embedded version of ZooKeeper, which is included in the product installation. If you use an external ZooKeeper server, it must be set up before you create the basic domain.

An *enterprise domain* can have multiple resources and users. This type of domain is typically used for production environments. You can configure high availability to ensure that Streams can continue to run even if resources fail or are not available. For high availability, you need to set up and configure an Apache ZooKeeper server for managing and storing configuration information. The method that you use to authenticate users in the domain must also be highly available. The type of enterprise domain that you create depends on your authentication mechanism. The following options are available as default authentication methods:

- Enterprise PAM domain: For high availability, use a Pluggable Authentication Module (PAM) with the LDAP backend. PAM must be set up and configured before you create the domain. The LDAP server must be accessible from the resource that is running the authentication and authorization services. There are other options that might not be highly available, such as PAM with a UNIX backend. For this option, the Streams users must be defined on this system.
- Enterprise LDAP domain: For high availability, use a Lightweight Directory Access Protocol (LDAP) server. LDAP must be set up and configured before you create the domain.

Related tasks:

Setting up a Streams basic domain

A basic domain provides a quick and easy method for using Streams with minimal setup. This type of domain is typically used for test or application development environments.

Setting up a Streams enterprise domain

An enterprise domain can have multiple resources and users. This type of domain is typically used for production environments.

Administering domains

1.2.2. Streams Instances

A *Streams instance* is a Streams runtime environment. It is composed of a set of interacting services running across one or more resources. Before you can use ¨Streams, you must create at least one instance.

Before you can create an instance, you must create a domain. Instances use the resources and security model that are configured in the domain. Each Streams instance operates as an autonomous unit and can be shared by multiple users. You can create multiple instances in a domain.

Even though instances can share resources, there is no logical interference with other instances. To avoid any physical interference, for example related to competing for the CPU cycles, you can allocate resources exclusively to instances.

Jobs are submitted to an individual instance, and no other instances are aware of the jobs. For example, you install Streams on a set of resources A, B, C, and D. You can configure instance1 to use resources A and B and configure instance2 to use resources B, C, and D. Jobs that you submit to instance1 are not aware of jobs that you submit to instance2, even though they share a common resource B.

The type of Streams environment determines many of the configuration decisions for Streams instances, for example:

Production environment

Provide a set of resources and network infrastructure that is solely allocated to the Streams instance and its applications.

Test and development environment

A set of resources can be shared by multiple instances that are used by individual users or a group of users. The implicit assumption in this case is that the users understand that no exclusivity requirement is met in terms of placing application components.

Related tasks:

Administering instances

Related information:

Considerations for Streams environments with multiple resources

The preferred and most reliable environment for multiple hosts is a production environment. If you do not need high availability, a development or test environment might be your preferred option.

1.2.3. Domain and instance services

When you start a Streams domain, instance, or stream processing application, services start on the appropriate resources.

You can manage which services run on specific resources by using tags. For example, you can apply the application tag to a resource so that it runs the application services. Likewise, you can apply the management tag to a resource so that it runs the domain and instance management services. If you prefer to run specific management services on specific resources, use the appropriate tag for that service (for example, `authentication`, `audit`, `jmx`, `sws`, or `view`).

Domain services

The following services are domain services:

Table 1.1. Domain Services

Domain service	Description
authentication and authorization service	The authentication and authorization service is a management service that authenticates and authorizes operations for the domain and all of the instances in the domain.

Domain service	Description
	If you want this service to run on specific resources, use the following tags: `authentication`. The following short name is used for this service in messages and commands: `aas`.
domain controller service	A domain controller service runs on every resource in a Streams domain and manages all of the other services on that resource. This service starts when you configure the resource in the domain, or when you run the `streamtool registerdomainhost` or `streamtool startdomainhost` command. This service is not associated with any tags. For more information, see Options for setting up the domain controller service on resources. The following short name is used for this service in messages and commands: `controller`.
logging service	The logging service is a management service that has two functions. It writes user messages in the streams.0.log file. It is also involved in audit logging. For more information, see Audit logging support for Streams. If you want this service to run on specific resources, use the following tags: `audit`. The following short name is used for this service in messages and commands: `auditlog`.
management API service	The management API service is a management service that provides an application programming interface (API). The Streams Console and Streams Studio use this API to obtain information about the domain. There can be only one management API service per resource. If you want this service to run on specific resources, use the following tags: `jmx`. The following short name is used for this service in messages and commands: `jmx`.
web management service	The web management service is a management service that provides web-based access to instance services. There is only one web management service in a domain. If you want this service to run on specific resources, use the following tag: `sws`. The following short name is used for this service in messages and commands: `sws`.

Instance services

The following services are instance services:

Table 1.2. Instance Services

Instance service	Description
application deployment service	The application deployment service is an application service that prepares application resources for deploying and running processing elements. It manages the deployment of application bundles to the appropriate resources. If you want this service to run on specific resources, use the `application` tag. This service runs on every application resource in the instance. The following short name is used for this service in messages and commands: `app`.
application manager service	The application manager service is a management service that administers the applications in the instance. This service handles job management tasks, including user requests such as job submission and cancellation. It interacts with the scheduler to compute the placement of processing elements that are associated with an application. It also interacts with the domain controller service to deploy and cancel the execution of these processing elements. If you want this service to run on specific resources, use the `management` tag. The following short name is used for this service in messages and commands: `sam`.
application metrics service	The application metrics service is a management service that initializes the Streams instance and monitors all instance services. The application metrics service collects run time metrics on the resources and Streams components. This service also collects performance metrics that are necessary for scheduling and system administration by interacting with the domain controller service. If you want this service to run on specific resources, use the `management` tag. The following short name is used for this service in messages and commands: `srm`.
Processing elements (PEs) and processing element container services (PEC)	Streams Processing Language (SPL) programs consist of operators, which are grouped at compile time into partitions. The execution container for a partition is a processing element (PE). Each partition runs in one PE. PEs are loaded by the processing element container service, which is started and monitored by the domain controller service for the resource that the PE is running on. If you want this service to run on specific resources, use the `application` tag.
view service	The view service is a management service that administers the data that is used in views. A view defines the set of attributes that can be displayed in a chart or a table. For more information, see “Views, charts, and tables”. If there are many data visualization users in your instance, consider running this service on a separate resource from other management services.

Instance service	Description
	If you want this service to run on specific resources, use the `view` tag. The following short name is used for this service in messages and commands: `view`.

Related tasks:

Administering services

1.2.4. Streams Interfaces

Streams provides a command line interface and the following graphical user interfaces: Streams Console and Streams Studio.

1.2.4.1. Streams Console

Streams includes an integrated console that you can use to view the health of the instances and applications in your domain. You can manage instances and resources, configure security, and monitor jobs from a single location.

Related tasks:

Starting the Streams Console

To open the Streams Console in your default web browser, you can use the `streamtool` launch command.

1.2.4.2. Streams Studio

Streams Studio is an integrated development environment (IDE) to create, edit, visualize, test, debug, and run SPL and SPL mixed-mode applications.

You can install Streams Studio on the Linux system where you install Streams. Alternatively, you can install it on a local Linux or Microsoft Windows system that accesses a remote Linux system where Streams is installed.

Related tasks:

Chapter 4, “Installing Streams Studio”

Use these procedures to install Streams Studio on a Linux or Microsoft Windows system.

Getting started with Streams Studio

Streams Studio includes the basic functions to create, configure, and monitor applications and includes sample applications and help information.

Installing Streams Studio for local or remote development

You can install Streams Studio on the Linux system where you install Streams, or on a local Linux or Microsoft Windows system that accesses a remote Linux system where Streams is installed.

Related information:

Introduction to Streams Studio

1.2.5. Resource managers

Streams includes a default resource manager, which allocates Streams resources. You can also use external resource managers, which run separately from Streams and allocate externally managed resources.

You can use both the Streams resource manager and an external resource manager together in a domain. Note that if you want to use an external resource manager in a domain, you must take extra steps to configure the domain to use it. You can also create your own external resource managers.

Related tasks:

Administering resource managers

Setting up an external resource manager for Streams

Streams supports user-defined resource managers. Using an external resource manager with Streams is optional.

1.2.6. Resources

Resources are physical and logical entities that Streams uses to run services.

You can obtain resources from the default Streams resource manager or an external resource manager.

Streams resources are always hosts. The externally managed resources might be other types of physical and logical entities, such as containers and process groups. Both Streams resources and externally managed resources can be used together in a domain.

You can allocate a resource to one or more domains. After you create instances in the domain, resources are allocated to the instances when the instances start (static) or at runtime (dynamically assigned). You can influence the allocation of resources by using tags.

You can configure resources to run specific types of Streams services by using tags. There are three general types of resources:

Application resource

A resource that runs application services and jobs only.

Management resource

A resource that runs management services only.

Mixed resource

A resource that runs application and management services and jobs.

By default, if multiple resources are allocated to a domain, Streams reserves one resource exclusively for management services.

Related tasks:

Administering resources

Setting up an external resource manager for Streams

Streams supports user-defined resource managers. Using an external resource manager with Streams is optional.

1.2.7. Stream processing applications

The main components of stream processing applications are tuples, data streams, operators, processing elements (PEs), and jobs.

Tuple

An individual piece of data in a stream. These data can be structured or unstructured. Typically, the data in a tuple represents the state of something at a specific point in time. For example, a stock ticker quote or a temperature reading from an individual sensor.

Data Stream

The running sequence of tuples.

Operator

An SPL operator manipulates the tuple data from the incoming stream and sends the results to zero or more output streams. Multiple streams and operators that are deployed in Streams form a *dataflow graph*.

Application Bundle

A file containing all logic (Operators, Data streams, Tuple definitions) and artifacts needed to run the application in a Streams instance. Application bundles are generated by the SPL compiler.

Job

A running stream processing application. Application bundles submitted to a Streams instance by Streams Studio or the **`streamtool submitjob`** command create and start jobs.

Processing Element (PE)

A Linux process that executes one or more operator logics, communicating data streams to other PEs as needed. A job consists of one or more PEs.

Related concepts:

Basics of streams processing applications

1.2.8. Views, charts, and tables

A view defines the set of attributes that can be displayed in a chart or table for a specific viewable data stream.

A *view* is the data stream that is produced by an operator output port that is has an `@view` annotation defined or that is defined in the Streams Console. When a view is started, it buffers the stream data for an active job. If you later cancel the job, the view can be restarted when another job is submitted for the same application.

You can optionally specify how much historical data to buffer in the view, when to start buffering data, and create a filter to restrict the tuple data that is kept in the view. With the Streams REST API, you can retrieve view information by using the REST API view collection resources.

After you create a view, you can use the buffered view data to create and save charts and tables. A chart can be used to graphically visualize numeric attributes only. A table can include numeric attributes and other types of data such as strings and timestamps. In Streams Studio, you can view streaming data in a table format only. With the Streams REST API, you can access the buffered tuple data from custom applications to build custom charts and tables by using third-party tools.

Related tasks:

Viewing streaming data in Streams Studio

In Streams Studio, you can use the properties view for an output port to show data that is flowing through a streams processing application.

Supported SPL data types for views, charts, and tables

Streams supports only certain Streams Processing Language (SPL) data types in views, charts, and tables.

Streams REST API overview

Streams provides an application programming interface (API) that you can use to access information about the configuration and health of installations, instances, and jobs.

Views

The REST API view collection resource provides access to information about all of the views that are associated with active jobs.

View items

The REST API view items collection resource provides access to the tuple data that is buffered by a view.

Part II. Installing Streams

Before installing Streams, review the planning roadmap. This roadmap summarizes important planning considerations, options, and requirements for the product. After installing Streams, you can install Streams Studio and other optional software to extend the functionality of Streams.

Chapter 2. Planning to install Streams

To plan for installing Streams, start by reviewing the information in the planning roadmap.

2.1. Preparing to install Streams

Use the information in this topic to plan and prepare for your Streams installation. The plan that you create for your environment determines your tasks and choices throughout the installation process.

Before you begin

These preparation tasks apply only to new installations of Streams. To prepare to upgrade your Streams environment, see the following information:

Chapter 5, “Upgrading Streams”

Decide what kind of Streams environment best suits your needs. The preferred and most reliable environment for multiple resources is a production environment. Streams also supports a highly available hybrid environment that uses Secure Shell (SSH). If you do not need high availability, a development or test environment might be your preferred option.

Use the environment considerations to make planning decisions on the following installation details:

- Installing as the root vs. non-root user
- Running management or application services on separate resources
- Setting up ZooKeeper on a dedicated resource

To review the considerations and make decisions for each environment type, see the following information: “Considerations for Streams environments with multiple resources”

RHEL 8 Compatibility

If you are planning to use Streams with Red Hat Enterprise Linux (RHEL) 8, consider the following:

- Use the Streams RHEL 7 bundle to install on RHEL 8. Additional RPM will be identified to allow Streams applications compiled on RHEL 7 to run on RHEL 8.
- All Streams applications intended for use on RHEL 8 *must* be compiled on RHEL 7 first.
- One domain cannot have resources for both RHEL 7 and RHEL 8.

Procedure

1. Set up the hardware that you need for all of the resources in your environment. A Streams resource refers to an entity on which streams processing applications and Streams services can be run, for example a host.

   Verify that your resources meets the hardware requirements for Streams by using the following information: “Version management for Streams”

2. Configure the operating systems on your Streams resources. Dependencies and restrictions exist for some of the operating system choices. Verify that your target systems meet the operating system requirements by using the following information: “Operating system requirements for Streams”

3. On every resource, ensure that the character encoding of your locale configuration is set to UTF-8.

4. On every resource where you want to use a Streams interface, install an X Window System. An X Window System is required to use the Streams Interactive GUI installation method, Streams Console, and Streams Studio on Linux.

5. Determine whether your environment requires additional software. The required Java environment is included in the installation package. Use the following information to determine whether you need additional software prerequisites:

   - "Required RPMs for Streams"

6. Download the Streams installation files.

   For information about downloading the current version of Streams, see Downloading Streams.

   The download package includes the following components:

   - Installation dependency checker tool
   - Java for application development

7. After you download the installation package to your system, use the dependency checker to find and fix any issues with your resources and environment.

   The Streams installation package provides a script that verifies whether your environment meets the requirements for Streams. Use the instructions at the following link to run the checking tool: "Streams dependency checker script"

8. Fix any dependency issues and optimize the environment.

   The dependency checker provides results that you can use to prepare and fix your Streams environment before installation. Use the information in the following sections to determine the correct updates and settings for your resources:

   - "Streams name resolution requirements for resources": All resources must satisfy the product network configuration requirements.
   - "Options for setting up the domain controller service on Streams resources: You can set up the domain controller service as a registered Linux system service or as an unregistered service.
   - "Firewall configuration guidelines for Streams": These guidelines apply to resources that are in the Streams cluster as well as clients that are external to the cluster.
   - "Guidelines for configuring Linux ulimit settings for Streams": Use these guidelines to ensure that your ulimit settings are sufficient for Streams.
   - "Performance considerations and options for Streams": There are several features and configuration options that might improve the performance of Streams.

What to do next

After you prepare the environment and install the prerequisite software, use the installation tools that are provided in the package to install Streams. For complete instructions, see the following information: Chapter 3, "Installing the Streams product"

Related tasks:

Chapter 3, “Installing the Streams product,”

Use these procedures to install Streams Version 7.1.0. If a previous version of Streams is already installed, use the procedures in Upgrading to Streams Version 7.1.0 .

Related reference:

“Streams dependency checker script”

The dependency checker script verifies that your system satisfies the operating system, RPM, and other software requirements for Streams.

Streamtool commands

This reference section provides a description of each `streamtool` command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

Resources

2.2. Considerations for Streams environments with multiple resources

Review these options and requirements for Streams environments before you install the product.

2.2.1. Considerations for a Streams production environment

The preferred and most reliable environment for multiple resources is a production environment. This option provides high availability so that Streams continues to run even if resources fail or are not available.

Installation

A root or non-root user can install the main installation package for Streams. This package includes all of the product files.

The domain host and resource installation packages contain a subset of the product files.

- If you plan to use the default Streams resource manager, install the domain host installation package on the resources. A root or non-root user can install this package.
- If you plan to use an external resource manager, install the resource installation package on the externally managed resources. A root user must install this package.

You can use the following options to install Streams:

- Install Streams on each resource. For high availability, install the main installation package on at least two resources and the domain host installation package or resource installation package on all additional resources.
- Use a shared file system, such as Network File System (NFS) or General Parallel File System (GPFS™), and install Streams in a shared directory that is accessible from each resource.

Domain controller service

For high availability, set up the domain controller service on all resources. For more information, see Options for setting up the domain controller service on resources .

Enterprise domain prerequisites

Set up an Apache ZooKeeper server and your user authentication method before you create a Streams enterprise domain.

- For managing and storing configuration information, Streams requires a ZooKeeper server.
- For high availability, you can use a Lightweight Directory Access Protocol (LDAP) server or Pluggable Authentication Module (PAM) with the LDAP backend for default user authentication. After creating the domain, you can use the following additional options to customize user authentication:
  - Login module authentication
  - Client certificate authentication
  - Kerberos authentication

Standby domain and instance services

Configure Streams domains and instances to have standby domain and instance services. For more information, see Configuring high availability for Streams enterprise domains and instances.

Related concepts:

"Streams installation packages"

Use the *main installation package* to install all of the product files on the resources in your Streams domain. To reduce the product size requirement on resources, the *domain host installation package* and *resource installation package* contain only a subset of the product files. For example, these installation packages do not include the toolkits.

Domain and instance services

See this information for descriptions of the Streams domain and instance services.

Related reference:

Prerequisites for an enterprise domain

2.2.2. Considerations for a Streams hybrid environment

Streams supports a highly available hybrid environment that uses Secure Shell (SSH).

The options and requirements for setting up a hybrid environment are similar to a production environment. This section describes the differences.

SSH

Each user of the product must enable SSH. For more information, see "Configuring a Secure Shell environment for Streams".

Installation

A root or non-root user installs the main installation package on all resources or uses a shared file system. This package includes all of the product files.

Notes:

- The preferred installation method in an SSH environment is to use a shared file system such as NFS or GPFS, and install Streams in a shared directory that is accessible from each resource.
- If you install Streams on each resource, you must satisfy the following requirements:
  - Install the product in the same location on each resource.
  - The installation owner and group must be the same on each resource.
  - The same version of the product must be installed on each resource.
  - If the installation is shared by multiple users, you must start the domain as the installation owner.
- If the installation is shared by multiple users, all users including the installation owner must be part of the group that owns the installation.

Domain controller service

A root or non-root user can set up the domain controller service as a registered Linux system service or an unregistered service. For more information, see “Options for setting up the domain controller service on Streams resources”

Related concepts:

“Considerations for a Streams production environment”

The preferred and most reliable environment for multiple resources is a production environment. This option provides high availability so that Streams continues to run even if resources fail or are not available.

“Streams installation packages”

Use the *main installation package* to install all of the product files on the resources in your Streams domain. To reduce the product size requirement on resources, the *domain host installation package* and *resource installation package* contain only a subset of the product files. For example, these installation packages do not include the toolkits.

Domain and instance services

See this information for descriptions of the Streams domain and instance services.

2.2.3. Considerations for Streams development and test environments

If you do not need high availability with automatic recovery from failures, a development or test environment might be your preferred option. Using Secure Shell (SSH) is optional.

Development or test environment without SSH

- A non-root user installs Streams on each resource or uses a shared file system.
  - If Streams is installed on each resource, you can use either of the following options:
    - Install the main installation package on at least one resource and the domain host installation package on all additional resources.
    - Install the main installation package on all resources.

    The main installation package includes all of the product files. The domain host installation package contains a subset of the product files.
  - If a shared file system such as NFS or GPFS is used, install Streams in a shared directory that is accessible from each resource.
- Set up the domain controller service on all resources. For more information, see "Options for setting up the domain controller service on Streams resources".
- Set up an Apache ZooKeeper server and your user authentication method before you create a Streams enterprise domain.
  - For managing and storing configuration information, Streams requires a ZooKeeper server.
  - For high availability, you can use a Lightweight Directory Access Protocol (LDAP) server or Pluggable Authentication Module (PAM) with the LDAP backend for default user authentication. After creating the domain, you can use the following additional options to customize user authentication:
    - Login module authentication
    - Client certificate authentication
    - Kerberos authentication

Development or test environment with SSH

- Each user of the product must enable SSH. For more information, see "Configuring a Secure Shell environment for Streams"
- A non-root user installs the main installation package on all resources or uses a shared file system. This package includes all of the product files.

Notes

- The preferred installation method in an SSH environment is to use a shared file system such as NFS or GPFS, and install Streams in a shared directory that is accessible from each resource.
- If you install Streams on each resource, you must satisfy the following requirements:
  - Install the product in the same location on each resource.
  - The installation owner and group must be the same on each resource.
  - The same version of the product must be installed on each resource.

    - If the installation is shared by multiple users, you must start the domain as the installation owner.

  - If the installation is shared by multiple users, all users including the installation owner must be part of the group that owns the installation.

- Set up an Apache ZooKeeper server and your user authentication method before you create a Streams enterprise domain.

  - For managing and storing configuration information, Streams requires a ZooKeeper server.

  - For high availability, you can use a Lightweight Directory Access Protocol (LDAP) server or Pluggable Authentication Module (PAM) with the LDAP backend for default user authentication. After creating the domain, you can use the following additional options to customize user authentication:

    - Login module authentication

    - Client certificate authentication

    - Kerberos authentication

Related concepts:

"Streams installation packages"

Use the *main installation package* to install all of the product files on the resources in your Streams domain. To reduce the product size requirement on resources, the *domain host installation package* and *resource installation package* contain only a subset of the product files. For example, these installation packages do not include the toolkits.

Domain and instance services

See this information for descriptions of the Streams domain and instance services.

Related reference:

Prerequisites for an enterprise domain

2.3. Hardware requirements for Streams

Before you install Streams, ensure that the target system satisfies the minimum hardware requirements.

Table 2.1. Minimum hardware requirements for Streams

Component	Minimum requirements	Comments
System	x86-64 (64-bit)	Streams supports Red Hat Enterprise Linux (RHEL).
Display	1280 x 1024	Lower resolutions are supported but not preferred for Streams Studio.

Component	Minimum requirements	Comments
Memory	2 GB	The amount of memory that is required by Streams is dependent on the applications that are developed and deployed. This minimum requirement is based on the memory requirements of the Commodity Purchasing sample application and other samples that are provided with the product.
Disk space	If installing the main installation package: ● 7 GB for the installation package and extraction ● 4 GB for the /tmp directory or other temporary directory, if specified by setting the IATEMPDIR environment variable. This space is cleaned up and available after the installation completes. ● 8.5 GB for the installation directory	Includes disk space required for installation and development resources only. Additional disk space is needed to run Streams. The amount depends on how you set up and configure your Streams environment. For more information about using the IATEMPDIR environment variable to configure a different temporary directory for installation processing, see the Preinstallation roadmap.
	If installing the domain host installation package or resource installation package: ● 1.2 GB for the installation package and extraction ● 400 MB for the /tmp directory or other temporary directory, if specified by setting the IATEMPDIR environment variable. This space is cleaned up and available after the installation completes. ● 1.5 GB for the installation directory	

2.4. Software requirements for Streams

Before you install Streams, run the dependency checker script to verify that the target system satisfies the product software requirements.

Related reference:

"Software prerequisites for Streams Studio"

Before you install Streams Studio, ensure that you satisfy the prerequisite software requirements.

2.4.1. Streams dependency checker script

The dependency checker script verifies that your system satisfies the operating system, RPM, and other software requirements for Streams.

2.4.1.1. Running the Streams dependency checker script

Run the dependency checker script before you install Streams to ensure that you satisfy all product requirements. You can also run this script after installation. In an environment that includes multiple resources, run this script on all resources and correct any incompatibilities before you use Streams.

Before you begin

To ensure that all product requirements are checked, the installation owner needs to run this script.

Procedure

- To run the dependency checker script before you install Streams:

  1. If you have not extracted the contents of the product installation package, enter the following command:

     ```
     tar -zxvf product-installation-package-name.tar.gz
     ```

     The **tar** command creates the StreamsInstallFiles directory, which contains the self-extracting binary file (StreamsSetup.bin) and other installation files.

  2. Change to the Streams installation files directory:

     ```
     cd product-installation-package-directory/StreamsInstallFiles
     ```

  3. To run the dependency checker script, enter the following command:

     ```
     ./dependency_checker.sh
     ```

- To run the dependency checker script after you install Streams, enter the following commands:

  ```
  cd product-installation-root-directory/7.1.0.0/bin
  ./dependency_checker.sh
  ```

  The following command options are optional:

  -v *level*

  Verbose option that displays additional messages. Possible values for *level* are:

  1. Includes additional informational messages.
  2. Includes additional debug messages.

  -h

  Display the help information.

Related reference:

“Dependency checker script output example for an x86_64 system running RHEL 7.9”

“Dependency checker script output example for an x86_64 system running RHEL 8.9”

Dependency checker script output example for an x86_64 system running RHEL 7.9

In the following example, the Default Java and Java Home sections show the Java settings that you have configured for your environment. For more information, see Java requirements, options, and settings for Streams.

Example 2.1. Dependency checker script output example for an x86_64 system running RHEL 7.9

```
$ ./dependency_checker.sh

21CS Streams 7.1.0.0 Dependency Checker
Date:  May 22, 2024 3:13:39 PM CDT
Edition:  21CS Streams

=== System Information ===
* Hostname:  dev1-rl7.teracloud.com
* IP address:  10.20.10.8
* Operating system:  Red Hat Enterprise Linux
* System architecture:  x86_64
* Security-Enhanced Linux setting:  Enforcing
* Default Java:
  * Java vendor:  Red Hat, Inc.
  * Java version:  1.8.0_382
  * Java VM version:  25.382-b05
  * Java runtime version:  1.8.0_382-b05
  * Java full version:  NOT_SET
  * Java system encoding:  NOT_SET
* Java Home:  NOT_SET
* Encoding:  UTF-8
* User Limit (ulimit -S) Settings:
  * Max processes (-u):  100000
  * Open files (-n):  100000
* User Limit (ulimit -H) Settings:
  * Max processes (-u):  100000
  * Open files (-n):  100000

=== System Configuration Check ===
* Status:  PASS - Check:  IP address check
* Status:  PASS - Check:  Host name resolution check
* Status:  PASS - Check:  Operating system version and architecture check
* Status:  PASS - Check:  Encoding check
* Status:  PASS - Check:  User limit (ulimit -S) settings check
* Status:  PASS - Check:  User limit (ulimit -H) settings check
* Status:  PASS - Check:  Firewall service check

=== Software Dependency Package Check ===
* Status:  CORRECT VERSION - Package:  bash, System Version:  4.2.46-35.el7_9
* Status:  CORRECT VERSION - Package:  bzip2-libs, System Version:  1.0.6-13.el7
* Status:  CORRECT VERSION - Package:  chkconfig, System Version:  1.7.6-1.el7
* Status:  CORRECT VERSION - Package:  coreutils, System Version:
 8.22-24.el7_9.2
* Status:  CORRECT VERSION - Package:  diffutils, System Version:  3.3-6.el7_9
* Status:  CORRECT VERSION - Package:  gawk, System Version:  4.0.2-4.el7_3.1
* Status:  CORRECT VERSION - Package:  gcc, System Version:  4.8.5-44.el7
* Status:  CORRECT VERSION - Package:  gcc-c++, System Version:  4.8.5-44.el7
* Status:  CORRECT VERSION - Package:  glibc, System Version:  2.17-326.el7_9
* Status:  CORRECT VERSION - Package:  glibc-common, System Version:
 2.17-326.el7_9
* Status:  CORRECT VERSION - Package:  grep, System Version:  2.20-3.el7
```

```
* Status:  CORRECT VERSION - Package:  initscripts, System Version:
 9.49.53-1.el7_9.1
* Status:  CORRECT VERSION - Package:  libcap, System Version:  2.22-11.el7
* Status:  CORRECT VERSION - Package:  libstdc++, System Version:  4.8.5-44.el7
* Status:  CORRECT VERSION - Package:  make, System Version:  3.82-24.el7
* Status:  CORRECT VERSION - Package:  numactl-libs, System Version:
 2.0.12-5.el7
* Status:  CORRECT VERSION - Package:  openldap, System Version:
 2.4.44-25.el7_9
* Status:  CORRECT VERSION - Package:  openssh-clients, System Version:
 7.4p1-23.el7_9
* Status:  CORRECT VERSION - Package:  openssl, System Version:  1.0.2k-26.el7_9
* Status:  CORRECT VERSION - Package:  pam, System Version:  1.1.8-23.el7
* Status:  CORRECT VERSION - Package:  perl, System Version:  5.16.3-299.el7_9
* Status:  CORRECT VERSION - Package:  perl-Time-HiRes, System Version:
 1.9725-3.el7
* Status:  CORRECT VERSION - Package:  perl-XML-Simple, System Version:
 2.20-5.el7
* Status:  CORRECT VERSION - Package:  pkgconfig, System Version:  0.27.1-4.el7
* Status:  CORRECT VERSION - Package:  procps-ng, System Version:  3.3.10-28.el7
* Status:  CORRECT VERSION - Package:  rpm, System Version:  4.11.3-48.el7_9
* Status:  CORRECT VERSION - Package:  sed, System Version:  4.2.2-7.el7
* Status:  CORRECT VERSION - Package:  tar, System Version:  1.26-35.el7
* Status:  CORRECT VERSION - Package:  util-linux, System Version:
 2.23.2-65.el7_9.1
* Status:  CORRECT VERSION - Package:  which, System Version:  2.20-7.el7
* Status:  CORRECT VERSION - Package:  xdg-utils, System Version:
 1.1.0-0.17.20120809git.el7

=== Summary of Errors and Warnings ===

CDISI0003I The dependency checker evaluated the system and did not find errors
 or warnings.
```

Dependency checker script output example for an x86_64 system running RHEL 8.9

In the following example, the Default Java and Java Home sections show the Java settings that you have configured for your environment. For more information, see Java requirements, options, and settings for Streams.

Example 2.2. Dependency checker script output example for an x86_64 system running RHEL 8.9

```
$ ./dependency_checker.sh

21CS Streams 7.1.0.0 Dependency Checker
Date:  May 22, 2024 1:26:40 PM MST
Edition:  21CS Streams

=== System Information ===
* Hostname:  dev10-rl8.teracloud.com
* IP address:  10.20.20.14
* Operating system:  Red Hat Enterprise Linux 8.9 (Ootpa)
* System architecture:  x86_64
* Security-Enhanced Linux setting:  Disabled
* Default Java:
  * Java vendor:  IBM Corporation
  * Java version:  1.8.0_381
  * Java VM version:  2.9
  * Java runtime version:  8.0.8.11 - pxa6480sr8fp11-20230901_01(SR8 FP11)
```

```
  * Java full version:  8.0.8.11 - pxa6480sr8fp11-20230901_01(SR8 FP11)
JRE 1.8.0 Linux amd64-64-Bit Compressed References 20230817_56476 (JIT enabled,
 AOT enabled)
OpenJ9   - c425e1c
OMR      - e712c65
IBM      - 696e9df
  * Java system encoding:  UTF-8
* Java Home:  /opt/ibm/java-x86_64-80
  * Java vendor:  IBM Corporation
  * Java version:  1.8.0_381
  * Java VM version:  2.9
  * Java runtime version:  8.0.8.11 - pxa6480sr8fp11-20230901_01(SR8 FP11)
  * Java full version:  8.0.8.11 - pxa6480sr8fp11-20230901_01(SR8 FP11)
JRE 1.8.0 Linux amd64-64-Bit Compressed References 20230817_56476 (JIT enabled,
 AOT enabled)
OpenJ9   - c425e1c
OMR      - e712c65
IBM      - 696e9df
  * Java system encoding:  UTF-8
* Encoding:  UTF-8
* User Limit (ulimit -S) Settings:
  * Max processes (-u):  100000
  * Open files (-n):  100000
* User Limit (ulimit -H) Settings:
  * Max processes (-u):  100000
  * Open files (-n):  100000

=== System Configuration Check ===
* Status:  PASS - Check:  IP address check
* Status:  PASS - Check:  Host name resolution check
* Status:  PASS - Check:  Operating system version and architecture check
* Status:  PASS - Check:  Encoding check
* Status:  PASS - Check:  User limit (ulimit -S) settings check
* Status:  PASS - Check:  User limit (ulimit -H) settings check
* Status:  PASS - Check:  Firewall service check

=== Software Dependency Package Check ===
* Status:  CORRECT VERSION - Package:  bash, System Version:  4.4.20-4.el8_6
* Status:  CORRECT VERSION - Package:  bzip2-libs, System Version:  1.0.6-26.el8
* Status:  CORRECT VERSION - Package:  chkconfig, System Version:  1.19.2-1.el8
* Status:  CORRECT VERSION - Package:  compat-openssl10, System Version:
 1.0.2o-4.el8_6
* Status:  CORRECT VERSION - Package:  coreutils, System Version:  8.30-15.el8
* Status:  CORRECT VERSION - Package:  diffutils, System Version:  3.6-6.el8
* Status:  CORRECT VERSION - Package:  gawk, System Version:  4.2.1-4.el8
* Status:  CORRECT VERSION - Package:  gcc, System Version:  8.5.0-20.el8
* Status:  CORRECT VERSION - Package:  gcc-c++, System Version:  8.5.0-20.el8
* Status:  CORRECT VERSION - Package:  glibc, System Version:  2.28-236.el8_9.12
* Status:  CORRECT VERSION - Package:  glibc-common, System Version:
 2.28-236.el8_9.12
* Status:  CORRECT VERSION - Package:  grep, System Version:  3.1-6.el8
* Status:  CORRECT VERSION - Package:  initscripts, System Version:
 10.00.18-1.el8
* Status:  CORRECT VERSION - Package:  libcap, System Version:  2.48-6.el8_9
* Status:  CORRECT VERSION - Package:  libnsl, System Version:
 2.28-236.el8_9.12
* Status:  CORRECT VERSION - Package:  libstdc++, System Version:  8.5.0-20.el8
* Status:  CORRECT VERSION - Package:  make, System Version:  4.2.1-11.el8
* Status:  CORRECT VERSION - Package:  ncurses-compat-libs, System Version:
 6.1-10.20180224.el8
* Status:  CORRECT VERSION - Package:  numactl-libs, System Version:
 2.0.16-1.el8
```

```
* Status:  CORRECT VERSION - Package:  openldap, System Version:  2.4.46-18.el8
* Status:  CORRECT VERSION - Package:  openssh-clients, System Version:
 8.0p1-19.el8_9.2
* Status:  CORRECT VERSION - Package:  openssl, System Version:  1.1.1k-12.el8_9
* Status:  CORRECT VERSION - Package:  pam, System Version:  1.3.1-27.el8
* Status:  CORRECT VERSION - Package:  perl, System Version:  5.26.3-422.el8
* Status:  CORRECT VERSION - Package:  perl-Time-HiRes, System Version:
 1.9758-2.el8
* Status:  CORRECT VERSION - Package:  perl-XML-Simple, System Version:
 2.25-1.el8
* Status:  CORRECT VERSION - Package:  pkgconf, System Version:  1.4.2-1.el8
* Status:  CORRECT VERSION - Package:  procps-ng, System Version:  3.3.15-14.el8
* Status:  CORRECT VERSION - Package:  rpm, System Version:  4.14.3-28.el8_9
* Status:  CORRECT VERSION - Package:  sed, System Version:  4.5-5.el8
* Status:  CORRECT VERSION - Package:  tar, System Version:  1.30-9.el8
* Status:  CORRECT VERSION - Package:  util-linux, System Version:
 2.32.1-43.el8
* Status:  CORRECT VERSION - Package:  which, System Version:  2.21-20.el8
* Status:  CORRECT VERSION - Package:  xdg-utils, System Version:  1.1.2-5.el8

=== Summary of Errors and Warnings ===

CDISI0003I The dependency checker evaluated the system and did not find errors
 or warnings.
```

Related reference:

"Guidelines for configuring Linux ulimit settings for Streams" The default Linux user limit (**`ulimit`**) values might be too small for some Streams environments. Use these guidelines to ensure that your **`ulimit`** settings are sufficient for Streams.

2.4.2. Operating system requirements for Streams

Streams supports Red Hat Enterprise Linux (RHEL).

Important:

- If you are running Streams across multiple resources, ensure that you satisfy the following requirements:
  - All resources must run on the same product version and release of Streams.
  - All resources must run on the same hardware system (x86_64) and system architecture (64-bit).
  - All resources must have the same major operating system version (Version 7 or 8).
- Streams provides tools that help you to verify this requirement on single and multiple resources:
  - Before installation, run the dependency checker script to verify requirements on the resources that you plan to install Streams on. If you are installing the product on a cluster of resources, run the dependency checker script on each resource in the cluster. For more information about this tool, see Streams dependency checker script.
  - After installation, you can use the **`streamtool checkhosts`**, **`checkdomainhosts`**, and **`checkinstancehosts`** commands to verify requirements on single or multiple resources. For more information about these commands, see Verifying requirements for Streams resources.

2.4.2.1. Supported operating system versions for Streams

All Streams hosts must run a supported version of Red Hat Enterprise Linux (RHEL).

The following table lists the supported operating system versions on x86_64.

Table 2.2. Supported operating system versions for Streams

Operating system	System hardware and architecture	Supported operating system versions
RHEL	x86-64	Version 7.0 or later
		Version 8.9 or later

"Restrictions for the Streams specialized toolkits"

Before using the Streams specialized toolkits, review these restrictions for Red Hat Enterprise Linux (RHEL).

2.4.3. Java requirements, options, and settings for Streams

Ensure that you use a Java development kit that is supported by Streams. If Java memory issues occur, Streams provides configuration properties that you can use to allocate more memory for the Java environment.

2.4.3.1. Supported Java development kits for application development

For application development, Streams supports the IBM SDK Java Technology Edition Version 8 and the Java SE Development Kit 8 from Oracle.

The IBM SDK is installed in the `product-installation-root-directory/7.1.0.0/ java` directory when you install the Streams product.

Notes:

- The supported Java development kits have been tested with Streams Studio and stream processing applications that include operators that were implemented using the Streams Java Operator API.
- If the *JAVA_HOME* environment variable is set, it must be set to a supported Java 8 implementation.
- If the *JAVA_HOME* environment variable is set at compile time, its value is used to run the JVM for Java operators at run time. Otherwise, the Java development kit in the *product-installation-root-directory*/7.1.0.0/java directory is used.

Related concepts:

Java Operator API overview

See this information to learn more about using the Streams Java Operator API to implement user-defined operators in Java.

Related reference:

"Software prerequisites for Streams Studio"

Before you install Streams Studio, ensure that you satisfy the prerequisite software requirements.

2.4.3.2. Configuration settings for Java memory issues

If Java out-of-memory errors occur when you use Streams, you can allocate more memory for the Java environment by increasing the maximum JVM size for Streams services and user interfaces.

A property value of *undefined* indicates that a default value is not set for the property.

Updating the JVM size for Streams services

If you are running the domain controller service as a registered Linux system service, the default maximum JVM size is 1024 megabytes. You can use the following *streamtool* commands to set the JVM size for the controller to a different value:

- **registerdomainhost**
- **chdomainhostconfig**
- **getdomainhostconfig**
- **rmdomainhostconfig**

For more information about these commands, enter

```
streamtool man command-name
```

For the other Streams services, you can either explicitly set the maximum JVM size, or allow Streams to select a value based on the setting of the **domain.jvmSizeComputationEnabled** property:

- If the **domain.jvmSizeComputationEnabled** property is set to the default value of true, Streams selects a maximum JVM size based on the system memory usage.
- If the **domain.jvmSizeComputationEnabled** property is set to false, Streams selects a maximum JVM size based on the system default size.

You can specify the JVM size for Streams services by using the following domain and instance properties:

- **controller.maximumJvmSize**=1024 (Default)
- **domainServiceContainer.initialJvmSize**=*undefined*
- **domainServiceContainer.maximumJvmSize**=*undefined*
- **instanceServiceContainer.initialJvmSize**=*undefined*
- **instanceServiceContainer.maximumJvmSize**=*undefined*

For more information about the controller and domain properties, enter

```
streamtool man domainproperties
```

For more information about the instance properties, enter

```
streamtool man properties
```

If the **security.runAsRoot** property is set to true, the authentication and authorization service and logging service can run independently. Use the following domain properties to specify the JVM size for these services:

- **aas.initialJvmSize**=*undefined*
- **aas.maximumJvmSize**=*undefined*
- **auditlog.initialJvmSize**=*undefined*
- **auditlog.maximumJvmSize**=*undefined*

For more information about these properties, enter

```
streamtool man domainproperties
```

Updating the JVM size for the Streams Console

For Streams Console out-of-memory errors, you can either explicitly set the maximum JVM size for the web management service, or allow Streams to select a value based on the setting of the **domain.jvmSizeComputationEnabled** property:

- If the **domain.jvmSizeComputationEnabled** property is set to the default value of true, Streams selects a maximum JVM size based on the system memory usage.
- If the **domain.jvmSizeComputationEnabled** property is set to false, Streams selects a maximum JVM size based on the system default size.

To increase the amount of memory that can be allocated to the Java environment for the web management service, use the following domain properties:

- **sws.initialJvmSize**=*undefined*
- **sws.maximumJvmSize**=*undefined*

For more information about these properties, enter

```
streamtool man domainproperties
```

For troubleshooting information, see Java out-of-memory error in the Streams Console .

Updating the JVM size for the streamtool command-line interface

For **streamtool** command errors, you can set the maximum JVM size by using the STREAMTOOL_MAX_JVM_SIZE environment variable, for example:

```
export STREAMTOOL_MAX_JVM_SIZE=size
```

Notes:

- The *size* that you specify cannot exceed the amount of memory that is available on the resource where you run **streamtool** commands. For a *size* of 256 megabytes, specify 256m. For a *size* of 1 gigabyte, specify 1g.
- If the STREAMTOOL_MAX_JVM_SIZE environment variable is not set, Streams selects a maximum JVM size based on the system default size.

For troubleshooting information, see Java out-of-memory error when using streamtool commands.

Related reference:

"Java cache for Streams streamtool command operations"

By default, Streams uses a 16 MB Java cache to speed up the performance of **`streamtool`** command operations.

2.4.4. Required RPMs for Streams

To operate correctly, Streams requires a set of prerequisite RPMs.

Important:

- Install all prerequisite RPMs before you install Streams. The installation program checks which RPMs you need to install, but the program does not install prerequisite RPMs. The installation program detects if prerequisite RPMs are not installed on the current host or if installed RPMs are at an incompatible version level. If required RPMs are missing during the installation, the program allows you to continue the installation or stop it. After the installation, the list of outstanding dependencies is recorded in the installation summary log.
- If you try to use Streams and the prerequisite RPMs are not installed, the product will not run correctly.

Related reference:

"Required RPMs for the Streams specialized toolkits"

Before using the Streams specialized toolkits, ensure that you satisfy any additional RPM requirements for the toolkits.

2.4.4.1. RPM minimum versions and availability for Streams

For most RPMs, the default operating system version meets the Streams requirements. The RPMs that are required for Streams are available in the operating system installation package and the product installation package.

Notes:

- Streams is tested with the minimum RPM versions that are listed in the following tables. Other RPM versions might work with the product, but they have not been tested.
- Running the Streams dependency checker script verifies that all the required RPMs are installed.
- In addition to the RPMs that are required for the Streams product, there are RPM requirements for several of the SPL specialized toolkits. The dependency checker script does not identify RPMs that are required for these toolkits. For more information, see "Required RPMs for the Streams specialized toolkits".

For more detailed information about required RPMs for the Streams product, see the table for your operating system version and hardware system:

- x86_64 systems
  - Required RPMs for RHEL 7

Required RPMs for RHEL 8.9 on x86-64 systems

The following table lists the RPMs that are required for running Streams applications on RHEL 8.9.

RPM name	RPM name	RPM name	RPM name
bash	glibc	numactl-libs	procps-ng
bzip2-libs	glibc-common	openldap	rpm
chkconfig	grep	openssh-clients	sed
compat-openssl10[1]	initscripts	openssl[3]	tar
coreutils	libcap	pam	util-linux
diffutils	libnsl[1]	perl	which
gawk	libstdc++	perl-Time-HiRes	xdg-utils
gcc[2]	make	perl-XML-Simple	
gcc-c++[2]	ncurses-compat-libs[1]	pkgconf	

Table notes:

1. These RPMs are required to run Streams RHEL 7 applications on RHEL 8.
2. For the gcc and gcc-c++ RPMs, Streams requires the minimum and maximum versions in the following table.

RPM name	Operating system version	Minimum RPM version	Maximum RPM version
gcc and gcc-c++	8.9	8.5.0-20	8.5.9999

3. For the openssl RPM, Streams requires a minimum version of 1.0.1e.

Required RPMs for RHEL 7 on x86-64 systems

The following table lists the RPMs that are required if you are running RHEL Version 7.0 or later on an x86-64 system.

Notes:

- All required RPMs in the following table are available in the operating system installation package.
- Unless otherwise indicated, the minimum RPM version that is required for Streams is the default operating system version.
- Unless otherwise indicated, there is no maximum RPM version that is required for Streams.

RPM name	RPM name	RPM name	RPM name
bash	glibc	openssh-clients	rpm
bzip2-libs	glibc-common	openssl[2]	sed
chkconfig	grep	pam	tar
coreutils	initscripts	perl	util-linux
diffutils	libcap	perl-Time-HiRes	which

RPM name	RPM name	RPM name	RPM name
gawk	libstdc++	perl-XML-Simple	xdg-utils
gcc[1]	make	pkgconfig	
gcc-c++[1]	openldap	procps-ng	

Table notes:

1. For the gcc and gcc-c++ RPMs, Streams requires the minimum and maximum versions in the following table.

RPM name	Operating system version	Minimum RPM version	Maximum RPM version
gcc and gcc-c++	7.0	4.8.2-16.el7	4.8.9999
	7.1	4.8.3-9.el7	
	7.2	4.8.5-4.el7	
	7.3	4.8.5-11.el7	
	7.4	4.8.5-16.el7	
	7.5	4.8.5-28.el7	
	7.9	4.8.5-44	

2. For the openssl RPM, Streams requires a minimum version of 1.0.1e.

2.5. Options for setting up the domain controller service on Streams resources

A domain controller service runs on every resource in a Streams domain and manages all of the other services on that resource. You can set up the domain controller service as a registered Linux system service or an unregistered service.

A Streams domain can include resources that run the domain controller service as a registered Linux system service and an unregistered service. Both options can be used together in a domain. However, you cannot use both options on a resource in the domain. You must run the controller on each resource as either a system service or an unregistered service.

Options

Streams provides the following options for setting up the domain controller service:

- For high availability with automatic recovery from failures, a root user must set up the domain controller service as a registered Linux system service on the resource. If a system crash or failure occurs, the controller is automatically restarted on the resource when the system is restarted.

  Notes:

  - If the domain is stopped, the controller continues to run.
  - Running the domain controller service as a system service is required if you plan to use the **`security.runAsRoot`** domain property.

- For high availability with limited automatic recovery from failures, a root or non-root user can set up the domain controller service as an unregistered service on the resource. If a system crash or failure occurs, the controller must be manually restarted on the resource after the system is restarted.

  Notes:

  - If the domain is stopped and the controller is running as user root, the controller continues to run.
  - If the domain is stopped and the controller is running as non-root, the controller shuts down.
    - If SSH is enabled, the controller is automatically restarted when the domain is restarted.
    - If SSH is not enabled, the user who restarts the domain must manually restart the controller on the resource.

  By default, the **`domain.sshAllowed`** property is set to true, and Streams attempts to use SSH for communication between resources in the domain. If you are not using SSH for your communication mechanism, setting this property to false prevents unnecessary communication attempts between resources and SSH error messages. You can set this property when you create the domain or after the domain is created by using the **`streamtool`** command-line interface.

Commands

The following table lists the **`streamtool`** commands that you use to set up the domain controller service on Streams resources.

Command	Description
`streamtool mkhostpkg`	This command generates a domain host installation package that you can use to add Streams resources to a domain. If you do not want the controller to run as a registered Linux system service, specify the `--unregistered` option on the command. If a non-root user runs the command with this option, the domain must exist and be started. The **`streamsdomainhostsetup.sh`** script file is in the domain host installation package. This script installs Streams on the resource, registers the resource in the domain, and starts the domain controller service on the resource. If you specify the `--unregistered` option on the **`streamtool mkhostpkg`** command, the controller starts as an unregistered service. Otherwise, the controller starts as a registered Linux system service.
`streamtool registerdomainhost`	This command registers the Streams resource in the domain, sets up the domain controller service as a registered Linux system service, and starts the controller on the resource.

Command	Description
	You can run the **streamtool registerdomainhost** command before or after you create a Streams enterprise domain. Example: `streamtool registerdomainhost -d domain-id --zkconnect host:port` **Notes:** ● You must have root authority to run this command. ● You must specify a domain name and the `--zkconnect` option on this command. The domain name that you specify does not need to exist before running the command. The `--zkconnect` option specifies the name of one or more host and port pairs for the configured external ZooKeeper ensemble. If you specify multiple host and port pairs, separate each pair with a comma. This value is the external ZooKeeper connection string. To obtain this value, you can use the **streamtool getzk** command.
streamtool startdomainhost	This command starts the domain controller service on a Streams resource. ● If you set up the domain controller service to run as a registered Linux system service by running the **streamtool registerdomainhost** command, the controller starts as a registered system service. ● To start the domain controller service as an unregistered service, specify the `--unregistered` option on the command. If a non-root user runs the command with this option, the domain must exist and be started.

Related concepts:

"Considerations for Streams environments with multiple resources"

Review these options and requirements for Streams environments before you install the product.

Domain and instance services

See this information for descriptions of the Streams domain and instance services.

Related reference:

Streamtool commands

This reference section provides a description of each **`streamtool`** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

2.6. Firewall configuration guidelines for Streams

If firewall usage is required, the preferred configuration is to set up a firewall at the perimeter of the Streams cluster to restrict network access to resources in the cluster but not communication between the resources. When any communication passes through a firewall, latency is introduced. These guidelines apply to clients that are in the Streams cluster as well as clients that are external to the cluster.

Tip:

When you configure a port for a service, the placement of the service can be controlled by host tags so that the ports are open only on the resources that are configured to run that service. For information about using host tags, see Assigning tags to resources in a domain.

Guidelines for clients that are in the Streams cluster

If your security plan requires a firewall on the resource or between resources, the following communications must be enabled between resources and must be blocked from unauthorized external access:

- If you are using a Secure Shell (SSH) environment for Streams, SSH communication between runtime resources.
- Communication between Streams management services, which is limited to ports in the local port range (TCP/IP port numbers automatically assigned by the host machine). You can use the port range configuration property for the domain to control the range.
- TCP communication between processing elements, which is limited to ports in the local port range.
- HTTPS connections between the web management service (SWS) and Streams interfaces such as the Streams Console. Each Streams domain that is running the SWS service requires a user-assigned HTTPS port.
- Connections between the management API service (JMX) and Streams interfaces such as the Streams Java Monitoring and Management Console, the Streams Console, and Streams Studio. Each Streams domain that is running the JMX service requires a user-assigned port.
- All communication protocols between applications and any systems, such as an external Apache ZooKeeper server or external analytics services.

Guidelines for clients that are external to the Streams cluster

The following communications must be enabled:

- The Streams Console and clients of the REST API require HTTPS connections to the web management service (SWS). The SWS service requires a user-assigned HTTPS port.
    - The SWS service (*sws.port*)

- Streams Studio and clients of the management API service (JMX) require SSL/TCP connections to the following services. These services require a user-assigned port. A static port selection is preferred.
  - The domain JMX service (*jmx.port*). For information about setting the jmx.port property, see Changing the secure port number for the management API service.
  - The instance JMX service (*sam.jmxPort*)
  - The instance view service (*view.port*)
- Streams Studio and clients of the management API service (JMX) require HTTPS connections to the following services. All of these services dynamically allocate HTTPS ports in the port range configuration property for the domain. You can also set them to a static value.
  - The domain JMX service (*jmx.httpPort*)
  - The instance JMX service (*sam.jmxHttpPort*)
  - The instance view service (*view.httpPort*)
- Streams Studio requires a Remote System Explorer (RSE) connection to the system where Streams is installed. For information about the specific ports that are used, see Configuring Streams Studio for remote development.

Related concepts:

Domain and instance services

See this information for descriptions of the Streams domain and instance services.

Related tasks:

Changing the secure port number for the Streams Web Service

By default, the SWS service runs on port 8443. You can specify a different port number when you create or update a domain.

Changing the secure port number for the management API service (JMX)

By default, the JMX service runs on port 9975. You can specify a port number when you create or update a domain.

Related reference:

Assigning tags to resources in a domain

You can specify the tags for a resource at domain creation time, when a resource is added to a domain, or any time thereafter.

Related information:

Conflicts requesting ports from the local port range

This Troubleshooting procedure provides information about how to review and update local port ranges.

2.7. Guidelines for configuring Linux ulimit settings for Streams

The default Linux user limit (**`ulimit`**) values might be too small for some Streams environments. Use these guidelines to ensure that your **`ulimit`** settings are sufficient for Streams.

If the **`ulimit`** values are too small for your Streams environment, issues can occur such as system startup, processing element (PE), and application development and submission failures. You can use **`ulimit`** settings to control the use of system resources.

There are two types of **`ulimit`** settings:

- The *hard limit* is the maximum value that is allowed for the soft limit. Any changes to the hard limit require root access.
- The *soft limit* is the value that Linux uses to limit the system resources for running processes. The soft limit cannot be greater than the hard limit.

The following hard and soft *ulimit* settings were used when testing Streams on RHEL:

- **`max user processes`** value = 100000
- **`open files`** value = 100000
- **`pending signals`** value = 100000

2.7.1. Streams users that require ulimit settings

Depending on your environment, *ulimit* values must be set for specific users on the systems that run Streams services.

Ensure that you set *ulimit* values for the Streams users that are listed in the following table and that those values satisfy the Streams *ulimit* requirements.

Table 2.3. Users that require ulimit settings

Environment	***Set ulimit value for:***
The domain controller service is running as a registered Linux system service. The domain controller service is running unregistered as root. External resource manager is used to manage resources.	• Root user • User who owns the installation files • User on the *instance.runAsUser* property, if specified **Note** By default, non-root users inherit the **`ulimit`** settings of the root user because Streams processes are launched by the root user. If a non-root user needs different **`ulimit`** settings, the administrator must explicitly specify those settings in the Linux security file, for example:

Environment	*Set ulimit value for:*

	`root soft nofile 100000`
	`root hard nofile 100000`
	`streamsuser soft nofile 111113`
	`streamsuser hard nofile 111113`

The domain controller service is running unregistered as non-root. SSH is used as the communication mechanism.	User who starts the domain

2.7.2. Verifying ulimit requirements for Streams

Streams requires that the hard and soft **`ulimit`** values for the maximum number of processes and open files are set to `100000` or higher. Streams provides tools that you can use to verify that these **`ulimit`** settings meet the product requirements.

Notes:

- If you run the domain controller service as a registered Linux system service or unregistered as user root, the controller automatically sets the hard and soft limit values to `100000`, if required. No change is made if your current values are `100000` or higher.
- If you run the domain controller service unregistered as non-root or use SSH as your communication mechanism, the controller sets the soft limit value to the hard limit value.
- If the hard and soft limits for the **`max user processes`** and **`open files`** values are less than `100000`, the Streams dependency checker script and the **`streamtool checkhosts`**, **`checkdomainhosts`**, and **`checkinstancehosts`** commands issue warning messages.

Procedure

To verify that the hard and soft **`ulimit`** settings for the maximum number of processes and open files are sufficient for Streams, you can use the Streams dependency checker script or the **`streamtool checkhosts`**, **`checkdomainhosts`**, and **`checkinstancehosts`** commands. If your **`ulimit`** settings are not sufficient, messages are displayed in the command output.

Notes:

- The **`checkdomainhosts`** and **`checkinstancehosts`** commands return **`ulimit`** settings for the user who is running the domain controller service, which is either user root or the user who starts the domain. The dependency checker script and the **`checkhosts`** command return the **`ulimit`** settings for the user who runs the command.
- For a dependency checker script example, see “Dependency checker script output example for an x86_64 system running RHEL 6.8”.
- To obtain the help information for a specific **`streamtool`** command, enter `streamtool man` *`command-name`*.

Related concepts:

"Options for setting up the domain controller service on Streams resources"

A domain controller service runs on every resource in a Streams domain and manages all of the other services on that resource. You can set up the domain controller service as a registered Linux system service or an unregistered service.

Related tasks:

"Running the Streams dependency checker script"

Run the dependency checker script before you install Streams to ensure that you satisfy all product requirements. You can also run this script after installation. In an environment that includes multiple resources, run this script on all resources and correct any incompatibilities before you use Streams.

Related reference:

Streamtool commands

This reference section provides a description of each **streamtool** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

2.7.3. Reviewing ulimit settings for Streams

Use these commands to review **ulimit** settings for Streams users and services.

Note

The **ulimit** command output shows the **ulimit** settings for the user who runs the commands. For Streams users that require ulimit settings, ensure that their settings satisfy the Streams **ulimit** requirements.

Procedure

Review **ulimit** settings.

- To review the hard **ulimit** settings, enter the following command:

  ```
  ulimit -aH
  ```

  Command output similar to the following example is displayed:

  ```
  core file size       (blocks, -c) unlimited
  data seg size        (kbytes, -d) unlimited
  scheduling priority          (-e) 0
  file size            (blocks, -f) unlimited
  pending signals              (-i) 100000
  max locked memory    (kbytes, -l) unlimited
  max memory size      (kbytes, -m) unlimited
  open files                   (-n) 100000
  pipe size         (512 bytes, -p) 8
  POSIX message queues  (bytes, -q) 819200
  real-time priority           (-r) 0
  ```

```
stack size          (kbytes, -s) unlimited
cpu time           (seconds, -t) unlimited
max user processes          (-u) 257262
virtual memory      (kbytes, -v) unlimited
file locks                  (-x) unlimited
```

- To review the soft **`ulimit`** settings, enter the following command:

  ```
  ulimit -aS
  ```

  Command output similar to the following example is displayed:

  ```
  core file size      (blocks, -c) 0
  data seg size       (kbytes, -d) unlimited
  scheduling priority         (-e) 0
  file size           (blocks, -f) unlimited
  pending signals             (-i) 100000
  max locked memory   (kbytes, -l) unlimited
  max memory size     (kbytes, -m) unlimited
  open files                  (-n) 100000
  pipe size         (512 bytes, -p) 8
  POSIX message queues  (bytes, -q) 819200
  real-time priority          (-r) 0
  stack size          (kbytes, -s) 10240
  cpu time           (seconds, -t) unlimited
  max user processes          (-u) 257262
  virtual memory      (kbytes, -v) unlimited
  file locks                  (-x) unlimited
  ```

- To review the **`ulimit`** settings for a Streams service, enter the following command as user root:

  ```
  cat /proc/PID/limits
  ```

  Run this command for each process on the resource. For example, you can find the domain controller service process ID (PID) by using the **`streamtool getdomainstate`** command. In the following example, the PID for the controller on host2.ibm.com is 28350.

  ```
  $ streamtool getdomainstate -d StreamsDomain --long --fmt %Nf
  domain: StreamsDomain State: STARTED
  Resource: host1.ibm.com Status: RUNNING Services: WAITING:controller(22133) RUNNING:sws(22444)                  Tags:
   Version: *4.2.0.0
  Resource: host2.ibm.com Status: RUNNING Services: RUNNING:aas(28511),auditlog(28511),controller(28350),jmx(13944) Tags:
   Version: *4.2.0.0
  ```

  The following example shows the **`ulimit`** values for the controller process:

  ```
  $ sudo cat /proc/28350/limits
  ```

Limit	Soft Limit	Hard Limit	Units
Max cpu time	unlimited	unlimited	seconds
Max file size	unlimited	unlimited	bytes
Max data size	unlimited	unlimited	bytes
Max stack size	8388608	unlimited	bytes
Max core file size	0	unlimited	bytes
Max resident set	unlimited	unlimited	bytes
Max processes	100000	100000	processes
Max open files	100000	100000	files

Limit	Soft Limit	Hard Limit	Units
Max locked memory	65536	65536	bytes
Max address space	unlimited	unlimited	bytes
Max file locks	unlimited	unlimited	locks
Max pending signals	7239	7239	signals
Max msgqueue size	819200	819200	bytes
Max nice priority	0	0	
Max realtime priority	0	0	
Max realtime timeout	unlimited	unlimited	us

2.7.4. Updating ulimit settings for Streams

There are several ways to change *ulimit* settings. The examples in this procedure show one way to change the hard and soft *ulimit* settings for Streams.

Before you begin

Before you change *ulimit* settings, contact your system administrator.

Procedure

1. Update the **`ulimit`** settings as user root.

   The following examples show one way to change the hard and soft settings for all users of a Streams resource.

   Notes:

   - If a `.conf` file in the following examples does not exist on your system, create the file using the same name in the example and then edit the file as shown in the example.
   - For RHEL, the files in the `/etc/security/limits.d` directory override the properties in the `/etc/security/limits.conf` configuration file.
     - By default, RHEL sets some **`ulimit`** values in the limits.d file and a default file in the /etc/security/limits.d directory for the maximum user processes.
     - The **`filenumber`** in the file names in the `/etc/security/limits.d` directory indicates which file takes effect. The values in the file with the highest number in the name of the file are used. For example, if the `/etc/security/limits.d` directory contains the `21-nofile.conf` and `22-nofile.conf` files, the values in the `22-nofile.conf` file are used.

   Examples:

   To change the **`open files`** value, use the procedure for your operating system.

   - On RHEL, add the following line to the `/etc/security/ limits.d/`*`filenumber`*`-nofile.conf` file as shown in the following example:

```
* - nofile open-files-value
```

To change the **max user processes** value, use the procedure for your operating system.

- On RHEL, add the following line to the `/etc/security/ limits.d/filenumber-nproc.conf` file as shown in the following example:

  ```
  * - nproc max-user-processes-value
  ```

To set the *hard stack* and *soft stack* values, use the procedure for your operating system.

Note

The soft stack size might be too low for your Streams applications, especially if your applications require a large stack size. If the soft stack size is too low, PE issues can occur. Common symptoms are that the PE starts and then terminates, the PE fails to start, the PE starts but fails to reach a healthy state, or the PE starts and reaches an unknown state.

- On RHEL, add the following lines to the /etc/security/ limits.d/*filenumber*-stack.conf file as shown in the following example:

  ```
  * hard stack unlimited
  * soft stack 20480
  ```

---

2. For the changes to take effect:
   - If Streams services and applications are not running, the Streams user logs out and then logs into the system.
   - If Streams services and applications are running, use the procedure for your environment:
     - If using SSH for your communication mechanism, or running the domain controller service unregistered as non-root:
       a. Stop the domain.
       b. Start the domain.
     - If running the domain controller service as a registered Linux system service, or unregistered as user root:
       a. On each resource in the domain, run the **streamtool stopdomainhost** command.
       b. Stop the domain.
       c. On each resource in the domain, run the **streamtool startdomainhost** command. If running the controller unregistered, specify the `--unregistered option`.
       d. Start the domain.
3. To verify that the **ulimit** settings are updated, use the procedure in "Reviewing ulimit settings for Streams".

Related concepts:

"Options for setting up the domain controller service on Streams resources"

A domain controller service runs on every resource in a Streams domain and manages all of the other services on that resource. You can set up the domain controller service as a registered Linux system service or an unregistered service.

Related reference:

Streamtool commands

This reference section provides a description of each **streamtool** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man command-name`.

Related information:

Troubleshooting PE startup and connection problems

If updating the ulimit settings does not fix your PE startup or connection problem, this information provides other suggested actions that might resolve the problem.

2.8. Performance considerations and options for Streams

There are several features and configuration options that might improve the performance of Streams.

Related concepts:

Performance improvements for streams processing applications

These guidelines and tips might improve the performance of your streams processing applications.

2.8.1. Java cache for Streams streamtool command operations

By default, Streams uses a 16 MB Java cache to speed up the performance of **streamtool** command operations.

This Java cache is located in the `/tmp/javasharedresources` directory. The cache name is `C270M3F0A64P_com.ibm.streams.streamtool_user-id_G31`.

Important

To improve *streamtool* performance, clear the *streamtool* Java cache whenever you install a new Streams release or fix pack.

To clear the cache, enter the following command:
`product-installation-root-directory/7.1.0.0/bin/clearstreamtoolcache`

Related reference:

"Configuration settings for Java memory issues"

See this information to learn more about the Streams configuration settings that you can use to allocate more memory to the Java environment.

2.8.2. Linux channel bonding option for Streams

Channel bonding is a Linux configuration option that might improve the performance of Streams by increasing bandwidth and providing redundancy.

To configure channel bonding, the administrator binds the network interface cards (NICs) together into a single channel using the bonding kernel module and a special network interface, called a channel bonding interface. You can configure channel bonding on all of the Red Hat Enterprise Linux (RHEL) versions that are supported by Streams.

For more information, see the documentation for your operating system.

Related information:

RHEL documentation website

See the information about channel bonding for your RHEL version.

SUSE documentation website

See the SLES information about network device bonding.

Boost Streams performance with Linux channel bonding

2.9. Streams name resolution requirements for resources

All resources must satisfy the product network configuration requirements or Streams fails to start on the resource.

Restriction

When you create a Streams instance or add resources to an instance, you cannot use localhost or any other host name or address that resolves to a loopback address (127.*.*.*).

Configured with the Streams application network properties

Each resource can have several interfaces. You can separate the application and management services network usage for Streams by using the **`domain.applicationNetwork`** and **`instance.applicationNetwork`** properties.

For more information, see Application network properties for Streams domains and instances.

Configured without the Streams application network properties

Each resource can have several interfaces, as long as the IP address of one interface matches an entry in the Linux name resolver. Streams uses the matching interface for all communication and ignores the other interfaces.

Ensure that all resources satisfy the following network configuration requirements:

- The name of the configured resource must match an entry in the Linux name resolver, for example: DNS, LDAP, Network Information Service (NIS), or `/etc/hosts`. The Linux name resolver entry is configured in the `/etc/nsswitch.conf` file. For configuration information, see your Linux documentation.
- The Linux name resolver result must match an ipv4 IP address of one of the local non-loopback interfaces.
- The host name to IP address resolution must be global. That is, for each resource in a domain, the Linux name resolver on the other domain resources must return the same host name to IP address mapping for the resource. For example, the host name hostA must map to the same IP address from every resource in a domain..

Note

With these requirements met, the Streams runtime system exclusively uses the IP address from the instance configuration for all communication. The runtime system's TCP and LLM_RUM_TCP PE transports also use that same IP address.

Verifying local host name to IP address requirements

Streams provides tools that help you to verify this requirement on single and multiple resources:

- Before installation, run the dependency checker script to verify requirements on the resources that you plan to install Streams on. If you are installing the product on a cluster of resources, run the dependency checker script on each resource in the cluster. For more information about this tool, see Streams dependency checker script.
- After installation, you can use the **`streamtool checkhosts`**, **`checkdomainhosts`**, and **`checkinstancehosts`** commands to verify requirements on single or multiple resources. For more information about these commands, see Verifying requirements for Streams resources.

Related information:

Configuring transport mechanisms

You can configure the transport mechanism by updating the application network properties for domains and instances, or by specifying options in stream processing applications.

2.9.1. Running Streams on a standalone system with no network connectivity

Use this procedure to run Streams on a standalone system with no network connectivity.

Before you begin

You must have root authority to complete this procedure.

Procedure

1. Load the network interface dummy, for example:

```
/sbin/modprobe dummy
```

The dummy network interface is a fake network adapter similar to loopback.

2. Assign `dummy0` a private address, for example:

```
/sbin/ifconfig dummy0 192.168.0.1
```

3. Obtain the host name and fully qualified host name by entering the following commands:

```
hostname
hostname -f
```

4. Add the following line to the /etc/hosts file by using your editor (for example, `vi/etc/hosts`):

```
192.168.0.1 host-name fully-qualified-host-name
```

5. As a non-root user, verify network connectivity by using the **`ping`** and **`ssh`** commands.

6. Create an instance that uses 192.168.0.1 as the only host.

2.10. Requirements and restrictions for the Streams specialized toolkits

Several of the specialized toolkits require RPMs that are not required by the Streams product. There are also toolkit restrictions on some platforms.

If you are upgrading to Version 7.1.0, there are also migration requirements for applications that affect several of the specialized toolkits.

2.10.1. Required RPMs for the Streams specialized toolkits

Before using the Streams specialized toolkits, ensure that you satisfy any additional RPM requirements for the toolkits.

- The Streams dependency checker script identifies any software dependencies that are required to run the Streams product. This script does not currently identify RPMs that are required for the specialized toolkits.
- Several toolkits require RPMs that are also required by the Streams product. If an RPM is required by the product, it is not listed for the toolkit.
- If you satisfy the RPM requirements for the Streams product, there are no additional required RPMs for many of the toolkits. There are additional required RPMs for the following toolkits:
  - Cybersecurity Toolkit
  - DPS Toolkit
  - Geospatial Toolkit
  - Internet Toolkit

  - Network Toolkit
  - TimeSeries Toolkit

- All required RPMs are available in the operating system installation package. The minimum required RPM version is the default operating system version. There is no maximum RPM version required.

Cybersecurity Toolkit

This section lists additional RPMs that are required for the Cybersecurity Toolkit. The following RPMs are required on all systems:

- bison
- flex
- libpcap

DPS Toolkit

This section lists additional RPMs that are required for the DPS Toolkit. The following RPMs are required on all systems:

- curl
- curl-devel
- lua
- lua-devel
- openldap-devel
- openssl-devel

Geospatial Toolkit

This section lists additional RPMs that are required for the Geospatial Toolkit PointMapMatcher operator.

To use the shared map feature of the PointMapMatcher operator, the following RPMs are required on all systems:

- boost
- boost-devel

Install version 1.55, or later.

Internet Toolkit

This section lists additional RPMs that are required for the Internet Toolkit. The following RPMs are required on all systems:

- libcurl

- libcurl-devel

Network Toolkit

This section lists an additional RPM that is required for the Network Toolkit.

The following RPM is required on all systems:

- libpcap

TimeSeries Toolkit

This section lists additional RPMs that are required for the TimeSeries Toolkit. The following RPMs are required on all systems:

- boost
- boost-devel
- libxml2
- libxml2-devel

Related reference:

"Restrictions for the Streams specialized toolkits"

Before using the Streams specialized toolkits, review these restrictions for Red Hat Enterprise Linux (RHEL).

"Streams dependency checker script"

See this information to learn more about the RPM checking performed by the dependency checker script.

"Required RPMs for Streams"

See this information to learn more about the required RPMs for the Streams product.

SPL specialized toolkits

This reference section includes documentation for all of the Streams specialized toolkits.

2.10.2. Restrictions for the Streams specialized toolkits

Before using the Streams specialized toolkits, review these restrictions for Red Hat Enterprise Linux (RHEL).

Table 2.4. Restrictions for the Streams specialized toolkits

Toolkit	Restriction
DPS Toolkit	This toolkit supports the following external data stores only: • Redis Version 2.8.2; Version 3.0, or later; Version 3.2, or later; and Version 4.0 or later

Related concepts:

“Supported operating system versions for Streams”

All Streams hosts must run a supported version of Red Hat Enterprise Linux (RHEL).

Related reference:

“Required RPMs for the Streams specialized toolkits”

Before using the Streams specialized toolkits, ensure that you satisfy any additional RPM requirements for the toolkits.

SPL specialized toolkits

This reference section includes documentation for all of the Streams specialized toolkits.

Chapter 3. Installing the Streams product

Use these procedures to install Streams Version 7.1.0. If a previous version of Streams is already installed, use the procedures in Upgrading to Streams Version 7.1.0.

3.1. Streams installation packages

Use the *main installation package* to install all of the product files on the resources in your Streams domain. To reduce the product size requirement on resources, the *domain host installation package* and *resource installation package* contain only a subset of the product files. For example, these installation packages do not include the toolkits.

If your Streams environment includes multiple resources, you must install the main installation package on at least one resource by using one of the following installation methods:

- Interactive graphical user interface (GUI) installation
- Interactive console installation
- Silent, response-file driven installation

After you install Streams on at least one resource, you can install a subset of the product files on the other resources in your enterprise domain by using the domain host installation package or resource installation package.

- If you plan to use the default Streams resource manager, install the domain host installation package. You can install this package on resources by using the automated installation method or when you set up an enterprise domain.
- If you plan to use an external resource manager, install the resource installation package. You can install this package on the externally managed resources when you set up an enterprise domain.

Alternatively, you can install the main installation package on each resource in the domain, or on a shared file system that is accessible by each resource in the domain.

Root or non-root users can install the main installation package and the domain host installation package. A root user must install the resource installation package.

If a Streams installation is shared by multiple users, you must install the product in a directory that can be accessed by all users. For installations that are shared by multiple users, the preferred method is to install as the root user and specify a non-personal administrator ID, or install by using a non-personal administrator ID such as streamsadmin.

The main differences between a root and non-root user installation are file ownership and the default location of the installation files. If you are running the installation program as a non-root user, you are the owner of the Streams installation. If you are running the installation program as a root user, the program prompts you to specify the user and group that will own the installed files. The root user cannot own the installed files.

Important

For both root and non-root installations, the user ID and group that own the installation must exist before running the Streams installation program.

During the installation, you are prompted to specify the root installation directory for the installed Streams files. The installation program appends a product version directory to the root directory. The default installation root directory depends on your authority:

- If you have root authority, the default root directory is `/opt/ibm/InfoSphere_Streams`.
- If you do not have root authority, the default root directory is *`home-directory`*`/InfoSphere_Streams`.

For example, if you are installing the product with root authority and accept the default installation directory, the product is installed in the `/opt/ibm/InfoSphere_Streams/7.1.0.0` directory.

Related tasks:

"Installing Streams by using the interactive or silent method" Use these procedures to install the Streams main installation package on a resource. This package includes all of the product files.

"Installing Streams by using the automated method"

The automated installation method enables you to build an installation package that you can use to install the Streams product on a large number of networked resources that share the same configuration.

Related reference:

Resources

3.2. Multiple installations of Streams on the same resource

To run multiple installations of Streams on the same resource, each installation must be installed in a unique location.

Restriction

If you install different versions of Streams on the same resource, separate versions of Streams Studio are required for each product version. If you install multiple installations of the same product version in different locations on a resource, you can use the same version of Streams Studio for all installations.

You can access multiple installations of Streams at the same time by using the **`streamsprofile.sh`** script. This script determines which installation of Streams you are using.

When you run the source *`product-installation-root-directory`*`/7.1.0.0/bin/`streamsprofile.sh command, your shell session uses that installation of Streams. To use another installation, you can source a different **`streamsprofile.sh`** script from your shell session.

If you are using a shell initialization file such as *`home-directory`*`/.bashrc` to source the **`streamsprofile.sh`** script, you can change the default installation that you use by editing the initialization file before you run the **`source`** command.

The configuration files for a Streams installation are stored in the *installation-owner-home-directory*/.streams directory. Included in this directory are license files, global configuration data, and instance data. Be careful with the global configuration data because its location is shared by all the Streams installations.

Installing a later version of Streams in the same location as a previous installation upgrades only that particular installation. No other installations on the same resource are affected.

Uninstalling Streams only affects that particular installation. This action does not affect other installations on the same resource.

Related concepts:

"Version management for Streams"

Related tasks:

"Configuring the Streams environment by running streamsprofile.sh"

3.3. Installing Streams by using the interactive or silent method

Use these procedures to install the Streams main installation package on a resource. This package includes all of the product files.

Before you begin

A response file is required for a silent installation. You can optionally use a response file for an interactive GUI or console installation.

About this task

The interactive installation methods guide you through the installation. The main difference between the interactive GUI and console methods is that an X Window System is required for a GUI installation. If an X Window System is installed, the default installation mode is interactive GUI. Otherwise, the default installation mode is interactive console.

The silent installation method enables standardized, repeatable installations across a number of resources. This method also enables you to install Streams as part of a larger suite or solution, or deploy the product using system management processes.

3.3.1. Preinstallation roadmap for the Streams interactive and silent installation methods

This roadmap summarizes preinstallation requirements and tasks for the Streams interactive and silent installation methods.

Table 3.1. Preinstallation roadmap for the interactive and silent installation methods

Task	Description
Review the installation considerations for Streams environments with multiple resources	The preferred and most reliable environment for multiple resources is a production environment. Streams also supports a highly available hybrid environment that uses Secure Shell (SSH). If you do not need high availability, a

Task	Description
	development or test environment might be your preferred option.
Create the user ID and group that will own the installation.	For both root and non-root installations, the user ID and group that own the installation must exist before running the Streams installation program. For installations that are shared by multiple users, the preferred method is to install as the root user and specify a non-personal administrator ID, or install by using a non-personal administrator ID such as `streamsadmin`. If you are running the installation program as a non-root user, you are the owner of the Streams installation. If you are running the installation program as a root user, the program prompts you to specify the user and group that will own the installed files. The root user cannot own the installed files.
Create a response file, if required.	A response file is required for a silent installation.
Update installation script files, if required	In Version 4.2, there are changes to several Streams installation file names.
Optional: Configure a different temporary directory for installation processing.	By default, the temporary directory used by the Streams installation program is the /tmp directory. To configure a different directory, update the *IATEMPDIR* environment variable by entering the following command: `export IATEMPDIR=`*`temporary-directory`* where *`temporary-directory`* is the path to the temporary directory, for example, `/home/streamsadmin`. **Note** If you do not have permission to execute files in the /tmp directory, the installation program might not run. To work around this issue, configure a different temporary directory.

Related reference:

"Sample response file for installing Streams"

The Streams sample response file contains values for all interactive and silent installation options.

3.3.2. Installing Streams by using the interactive GUI method

Use this procedure to install Streams by using the interactive GUI method.

Before you begin

Ensure that you satisfy the requirements in the Preinstallation roadmap.

Important

An X Window System is required for the interactive GUI installation.

Tip

If you reinstall your current release of Streams, you do not need to back up your user data because the installation location and the user data are in different locations.

Procedure

1. To extract the contents of the Streams installation package, enter the following command:

   ```
   tar -zxvf product-installation-package-name.tar.gz
   ```

   The **tar** command creates the `StreamsInstallFiles` directory, which contains the self-extracting binary file (StreamsSetup.bin) and other installation files.

2. To start the installation, run the Streams self-extracting binary file by entering the following commands:

   ```
   cd product-installation-package-directory/StreamsInstallFiles
   ./StreamsSetup.bin
   ```

3. Select the Streams edition that you are installing, and then click **Next**.

4. Review the information on the Introduction page, and then click **Next**.

5. Review the license agreement.

   - To continue the installation, click **I accept the terms in the license agreement** and click **Next**.
   - If you do not accept the license agreement, click **I do not accept the terms in the license agreement** and click **Cancel** to exit the installation program.

   ### Important

   You must accept the license agreement to continue the installation.

6. The installation program checks your system configuration for any incompatibilities with Streams and displays the results on the Software Dependencies page.

   ### Important

   - If an incompatibility is found, the installation program displays an error or warning message:
     - If an error is displayed, you cannot continue the installation until the error is fixed. For example, if you try to install 64-bit Streams on a 32-bit system, the installation cannot continue.
     - If a warning is displayed, you can continue or cancel the installation. If you continue, the installation completes successfully, but the incompatibilities must be corrected before you use Streams.

- The installation program displays information about required RPMs and their status on your system. The program also logs this information in the installation summary log.

  If any required RPM dependencies are not met, you have the following options:

  - Install or update the required RPM dependencies in another terminal window, select the installation program refresh option to verify that the required dependencies are now met, and then continue the installation.

  - Continue the installation without installing or updating the required RPM dependencies. You can complete the installation without correcting any RPM dependencies, but you must correct any issues before you use Streams.

  - Cancel and exit the installation.

  To save the dependency check results in a file, click **Save result to file**. To continue the installation, click **Next**.

---

7. If you are running the installation program as a root user, the program prompts you to specify the user and group that will own the installed files. The root user cannot own the installed files.

   If you are running the installation program as a non-root user, the Specify File Owner page is not displayed because you are the owner of the Streams installation.

8. On the Installation Directory page, click **Next** to accept the default root installation directory or enter the absolute path of another root directory.

   ## Important

   The installation directory cannot contain spaces or the following characters: ' " ` $ | & ? * < > \. If you include spaces or non-supported characters, the installation program displays an error message and requires you to enter a valid directory or cancel the installation.

   ---

9. The installation program displays the Preinstallation Summary page for your review before starting the installation. After reviewing this information, click *Install* to start the installation. The program displays the status of the installation.

10. Review the following options on the Postinstallation Tasks page:

    - To open a browser window that displays the *Release Notes* in HTML format, select **View release notes**.

    - To continue, click **Next**.

    The *Release Notes* are displayed in a browser window.

11. Click **Done** to exit the installation program.

12. If the installation program displayed warnings or errors during the installation, check the installation summary log file and resolve any issues before continuing. This log file is in the *`product-installation-root- directory`*`/7.1.0.0/logs` directory.

What to do next

- For a new installation of Streams, review the Postinstallation roadmap. This roadmap summarizes required and optional postinstallation tasks and options for Streams.
- If you installed the new version and have earlier versions installed, complete the procedure in Upgrading to Streams Version 7.1.0.
- If you installed a Version 7.1.0 fix pack or interim fix, complete the procedure in Upgrading Version 7.1.0.

3.3.3. Installing Streams by using the interactive console method

Use this procedure to install Streams by using the interactive console method.

Before you begin

Ensure that you satisfy the requirements in the Preinstallation roadmap.

Tip

If you reinstall your current release of Streams, you do not need to back up your user data because the installation location and the user data are in different locations.

Procedure

1. To extract the contents of the Streams installation package, enter the following command:

   ```
   tar -zxvf product-installation-package-name.tar.gz
   ```

   The **tar** command creates the `StreamsInstallFiles` directory, which contains the self-extracting binary file (`StreamsSetup.bin`) and other installation files.

2. To start the installation, run the Streams self-extracting binary file by entering the following commands:

   ```
   cd product-installation-package-directory/StreamsInstallFiles
   ./StreamsSetup.bin -i console
   ```

   ### Tips:

   - When you are instructed to enter a response, enter the response and then press Enter.
   - To change your response on a previous step, enter **back**.
   - To cancel the installation at any time, enter **quit**.

3. Select the Streams edition that you are installing, and then press Enter.
4. Review the information on the Introduction page, and then press Enter.
5. On the License Agreement page:

- To continue viewing the license agreement, press Enter.
- To accept the license agreement, enter `1`.
- To decline the license agreement, enter `2`.
- To print the license agreement, enter `3`.
- To return to the previous license agreement page, enter `99`.

Important

You must accept the license agreement to continue the installation. After you accept the license agreement, the installation starts.

6. The installation program checks your system configuration for any incompatibilities with Streams.

Important

- If an incompatibility is found, the installation program displays an error or warning message:
  - If an error is displayed, you cannot continue the installation until the error is fixed. For example, if you try to install 64-bit Streams on a 32-bit system, the installation cannot continue.
  - If a warning is displayed, you can continue or cancel the installation. If you continue, the installation completes successfully, but the incompatibilities must be corrected before you use Streams.
- If any required RPM dependencies are not met, the installation program displays this information. Information about all required RPMs and their status is logged in the summary log.

  If any required RPM dependencies are not met, you have the following options:
  - Install or update the required RPM dependencies in another terminal window, select the installation program refresh option to verify that the required dependencies are now met, and then continue the installation.
  - Continue the installation without installing or updating the required RPM dependencies. You can complete the installation without correcting any RPM dependencies, but you must correct any issues before you use Streams.
  - Cancel and exit the installation.

7. If you are running the installation program as a root user, the program prompts you to specify the user and group that will own the installed files. The root user cannot own the installed files.

   If you are running the installation program as a non-root user, the Specify File Owner page is not displayed because you are the owner of the Streams installation.

8. On the Installation Directory page, press Enter to accept the default root installation directory or enter the absolute path of another root directory.

Important

The installation directory cannot contain spaces or the following characters: ' " ` $ | & ? * < > \. If you include spaces or non-supported characters, the installation program displays an error message and requires you to enter a valid directory or cancel the installation.

9. The installation program displays the Preinstallation Summary page for your review before starting the installation. After reviewing this information, press Enter to continue.
10. When you are prompted to confirm the installation directory, press Enter to start the installation. The last line of the Installing message shows the status of the installation. When the installation completes, the Installation Complete page is displayed.
11. If the Installation Complete page indicates that the installation program detected errors or warnings, review the installation summary log file and resolve any issues. This log file is in the *product-installation-root- directory*`/7.1.0.0/logs` directory.

What to do next

- For a new installation of Streams, review the Postinstallation roadmap. This roadmap summarizes required and optional postinstallation tasks and options for Streams.
- If you installed the new version and have earlier versions installed, complete the procedure in Upgrading to Streams Version 7.1.0.
- If you installed a Version 7.1.0 fix pack or interim fix, complete the procedure in Upgrading Version 7.1.0.

3.3.4. Installing Streams by using the silent method

Use this procedure to install Streams by using the silent, response file-driven method.

Before you begin

Ensure that you satisfy the requirements in the Preinstallation roadmap.

Important

A response file is required for a silent installation. You must set the silent installation response file properties before you install Streams. For more information about creating a response file and setting the required properties, see Sample response file for installing Streams.

Tip

If you reinstall your current release of Streams, you do not need to back up your user data because the installation location and the user data are in different locations.

Procedure

1. To extract the contents of the Streams installation package, enter the following command:

```
tar -zxvf product-installation-package-name.tar.gz
```

The **tar** command creates the `StreamsInstallFiles` directory, which contains the self-extracting binary file (StreamsSetup.bin) and other installation files.

2. To start the installation, run the Streams self-extracting binary file by entering the following commands:

```
cd product-installation-package-directory/StreamsInstallFiles
./StreamsSetup.bin -i silent -f response-file
```

3. After the silent installation completes, check the installation summary log file to determine if the installation completed successfully. If any errors or warnings were issued during the installation, resolve any issues before continuing. This log file is in the *product-installation-root-directory*`/7.1.0.0/logs` directory.

What to do next

- For a new installation of Streams, review the Postinstallation roadmap. This roadmap summarizes required and optional postinstallation tasks and options for Streams.
- If you installed the new version and have earlier versions installed, complete the procedure in Upgrading to Streams Version 7.1.0.
- If you installed a Version 7.1.0 fix pack or interim fix, complete the procedure in Upgrading Version 7.1.0.

3.3.5. Sample response file for installing Streams

The Streams sample response file contains values for all interactive and silent installation options.

Attention:

- A response file is required if you use the silent installation method.
- A response file is optional if you use the interactive GUI or console installation method.
- You cannot use a response file to uninstall Streams.

The sample response file, `Streams_SampleResponseFile.properties`, is included in the Streams installation package. You can create a response file by copying the sample response file and adding your own settings to the property values. After Streams is installed, the sample response file is also in the *product-installation-root-directory*`/7.1.0.0/install` directory.

After you start the installation, the Streams installation program checks the *product-installation-package-directory*/StreamsInstallFiles directory for a response file with either of the following names:

- `Streams_SampleResponseFile.properties`
- `StreamsSetup.properties`

You can use a different name for your response file. If you use a different name, you must include the response file name on the installation command by using the `-f` option, for example:

```
./StreamsSetup.bin -i silent -f MyResponseFile.properties
```

Notes:

- If you created a response file in a previous Streams release, re-create the file by using the sample file in the installation package for the current release to ensure that you use the most current properties.
- In the following situations, the Streams installation program reads the response file even if the program is not running in silent mode:
  - The installation program finds a response file named `installer.properties` or `StreamsSetup.properties` in the *`product-installation-package- directory`*`/StreamsInstallFiles` directory.
  - You specify `-f` *`response-file-name`*`.properties` when you run the Streams installation command.

  If the installation program is running in interactive GUI or console mode, the program uses the values in the response file that apply to an interactive installation.
- If a response file property is not listed in the response file, the Streams installation program uses the default value, if applicable.

Response file properties

Note

Note: For more information about the response file properties, see the `Streams_SampleResponseFile.properties` file.

RESPONSE_FILE_VERSION

The value of this property is set by Streams and used by the installation program. If you change the value of this property, the installation might fail.

INSTALLER_LOCALE

Specifies the locale that Streams uses to run the installation and write to the installation log. In the current release, the only supported locale is `en` (English).

INSTALLER_UI

Specifies the installation mode and interface. If an X Window System is installed, the default mode for installing Streams is interactive, and the default interface is an interactive graphical user interface. Otherwise, the default mode is interactive console.

FILE_OWNER_GROUP

Identifies the group for the installed files. The default group is `streamsadmin`. This property applies only to installations that are run by a user with root authority.

FILE_OWNER_USER_ID

Identifies the user who owns the installed files. The default user ID is `streamsadmin`. This property applies only to installations that are run by a user with root authority.

USER_INSTALL_DIR (Optional)

Specifies the root installation directory for Streams. To override the default installation directory, uncomment this property and enter an absolute path of your choice. Read, write, and execute permissions are required for all existing directories in the specified path.

Important

The installation directory cannot contain spaces or the following characters: ' " ` $ | & ? * < > \. If the installation directory contains spaces or non-supported characters, the installation program cancels the silent installation.

IS_EDITION_TYPE (Silent installation only)

Specifies the Streams edition that you are installing. This property is required.

- To install the Streams product, specify Product.
- To install the Streams Developer Edition, specify Developer.

LICENSE_ACCEPTED (Silent installation only)

Verifies that the user who is installing Streams read and agreed to the license agreement.

Important

If this property is not set to TRUE, the silent installation fails. For instructions about how to access and review the license terms, see "Reviewing the license agreement for Streams".

VERSION_EXIST_ACTION (Silent installation only)

If you install the product in the same root installation directory, Streams installs each new product version or release in a unique directory. This property specifies the action to be taken if the installation program detects that you are installing the same version or release in the same root installation directory. By default, the installation program uninstalls the existing version and continues the installation.

MISSING_DEPENDENCY_ACTION (Silent installation only)

Specifies the action to be taken when the installation program detects missing dependencies. By default, the installation program continues the installation.

3.4. Postinstallation roadmap for Streams

The postinstallation roadmap summarizes required and optional postinstallation tasks and options for Streams.

Table 3.2. Postinstallation tasks for Streams

Task	Description
Configure the Streams environment variables.	Before you can use Streams, you must configure the product environment variables. For more information, see "Configuring the Streams environment by running streamsprofile.sh".
If you are upgrading to Streams Version 7.1.0 and have a previous version of the product installed, review the migration guidelines.	To review the migration guidelines, see this section.
Optional: Configure SSH for Streams.	In previous releases, Streams used Secure Shell (SSH) to run product applications, commands, and scripts. Beginning in Version 4.0, using SSH is optional. If you are using SSH, see Configuring a Secure Shell environment for Streams.
If you are not using SSH for your communication mechanism, you must set up the domain controller service on resources with the main installation package installed.	A domain controller service runs on every resource in a Streams domain and manages all of the other services on that resource. You can set up the domain controller service as a registered Linux system service or an unregistered service. For more information, see Options for setting up the domain controller service.
Set up a Streams domain.	Before you use Streams, you must create at least one domain. A Streams domain is a container for Streams instances which provides a single point for configuring and managing common resources, security, and instances. A *basic domain* has a single Streams resource and user. This type of domain is typically used for test or development environments. An *enterprise domain* can have multiple resources and users. This type of domain is typically used for production environments. You can configure high availability to ensure that Streams can continue to run even if resources fail or are not available. For more information, see the following procedures: ● Setting up a Streams basic domain on a single resource ● Setting up a Streams enterprise domain on multiple resources
Optional: Install Streams Studio.	Streams Studio is an integrated development environment that enables you to create, edit, visualize, test, debug, and run streams processing applications. You can install Streams Studio on a Linux or Microsoft Windows system. For information about installation requirements and procedures for Streams Studio, see Installing Streams Studio.
Optional: Install the Uncrustify source code formatter.	When you generate C++ code using the **sc** command or Streams Studio, Streams can use the Uncrustify source code formatter to ensure that the generated code is indented and formatted correctly. Installing and using the Uncrustify source

Task	Description
	code formatter with Streams is optional. For more information, see "Installing the Uncrustify source code formatter for Streams".
Optional: Change the default location for Streams class cache, log, and trace files.	The default root directory for class cache, log, and trace files that are generated by Streams is the `/tmp` directory. If you use a system maintenance utility such as **`tmpwatch`**, either modify the utility to exclude these files or configure a different root directory by using the following Streams domain properties: **domain.fileStoragePath** This property specifies the root directory where Streams class cache and other temporary files are stored. If the class cache files are removed, the web management service might fail and not recover correctly. **domainLog.path** This property specifies the root directory where Streams log and trace files are stored. If these files are removed, it might be difficult to resolve Streams problems. You can update these properties when you create the domain or after the domain is created.
Optional: Review your options for extending Streams.	You can expand the capability of Streams by using additional features. For more information, see Extending Streams.

3.4.1. Configuring the Streams environment by running streamsprofile.sh

Before you can use Streams, you must configure the product environment variables by running the **`streamsprofile.sh`** script that is included in the installation directory. This script is not automatically run by the installation program.

Procedure

To configure your local shell environment for Streams, enter the following command:

```
source product-installation-root-directory/7.1.0.0/bin/streamsprofile.sh
```

Notes:

- Add the **`source`** command to your *`home-directory/`*`.bashrc` shell initialization file. Otherwise, you must enter the command every time you start Streams. For example, if the product is installed in the `/home/streamsadmin/ InfoSphere_Streams/7.1.0.0` directory, add the following line to your `.bashrc` file:

  ```
  source /home/streamsadmin/InfoSphere_Streams/7.1.0.0/bin/
  streamsprofile.sh
  ```

- For shared installations, you can add the **source** command to the `/etc/profile.d` script so that all users do not need to edit their own shell initialization files. Users can modify their own shell initialization files if they need to use a different installation of Streams.
- If you install a new product version or a Version 7.1.0 fix pack or interim fix, you must update the `.bashrc` or `/etc/profile.d` file. Streams appends a unique version directory for a new product version, fix pack, or interim fix to the installation root directory.

Results

This script configures the following Streams environment variables:

- STREAMS_INSTALL=*product-installation-root-directory*/7.1.0.0
- STREAMS_INSTANCE_ID=StreamsInstance

> **Note**
>
> For compatibility with previous releases, if the STREAMS_DEFAULT_IID environment variable is already configured when you run the script, STREAMS_INSTANCE_ID is set to the value of STREAMS_DEFAULT_IID.

- STREAMS_DOMAIN_ID=StreamsDomain

This script also adds the *product-installation-root-directory*`/7.1.0.0/bin` directory to the PATH environment variable so that you can run **streamtool** commands without specifying a fully qualified path.

3.4.2. Configuring a Secure Shell environment for Streams

In previous releases, Streams used Secure Shell (SSH) to run product applications, commands, and scripts. Beginning in Version 4.0, using SSH is optional. If you are using SSH, you must configure a Secure Shell environment for Streams before you use the product.

Procedure

1. Set up your SSH keys.
   a. Enter **ssh-keygen -t dsa**.
   b. When prompted, enter a key file name or press Enter to accept the default name. See ❶ in the sample command output below.
   c. When prompted for a passphrase, press Enter twice without entering the passphrase. See ❷ and ❸ in the sample command output.

   The following example shows the *ssh-keygen* command output:

```
[bsmith@server44 ~]$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/bsmith/.ssh/id_dsa): ❶
Created directory '/home/bsmith/.ssh'.
```

```
Enter passphrase (empty for no passphrase): ❷
Enter same passphrase again: ❸
Your identification has been saved in /home/bsmith/.ssh/id_dsa.
Your public key has been saved in /home/bsmith/.ssh/id_dsa.pub.
The key fingerprint is:
6e:72:23:80:55:ec:73:d1:7f:10:09:16:a7:71:61:0d bsmith@server44.ibm.com
```

The **ssh-keygen** command creates an .ssh directory (*home-directory*/.ssh), if it does not exist, and two key files (id_dsa.pub for the public key and id_dsa for the private key).

2. Switch to the .ssh directory by typing the following command:

```
cd ~/.ssh
```

3. Append the public key file to the authorized_keys file by typing the following command:

```
cat id_dsa.pub >> authorized_keys
```

4. SSH requires that only the owner has access to the key files. Change the permission on the files in the .ssh directory by typing the following command:

```
chmod 600 *
```

SSH also requires that only the owner has write access to the .ssh directory. The **ssh-keygen** command creates this directory with the appropriate mode.

5. SSH requires that only the owner has write access to their home directory. To satisfy this requirement, type the following command:

```
chmod go-w ~
```

6. Test your SSH settings by using the **streamtool checkhosts** command.

3.4.3. Installing the Uncrustify source code formatter for Streams

When you generate C++ code using the *sc* command or Streams Studio, Streams can use the Uncrustify source code formatter to ensure that the generated code is indented and formatted correctly. Installing and using the Uncrustify source code formatter with Streams is optional.

If you specify the code beautification option (-z) when you compile your code, and the Uncrustify software is not installed, the following message is displayed:

```
CDISP9094W WARNING: The 'uncrustify' code formatter could not be found.
Compilation is continuing without formatting.
```

Procedure

1. Download the Uncrustify source code formatter from the Uncrustify website (http://uncrustify.sourceforge.net/).

2. Install the source code formatter by using the instructions on the Uncrustify website.

3. Edit the Uncrustify configuration file that is provided by Streams.

The uncrustify.verbose.cfg file is included in the Streams installation. You can edit this file to customize the formatting. This file is in the

product-installation-root-directory`/7.1.0.0/etc/uncrustify` directory. For more information about modifying the configuration file, go to the Uncrustify website.

Note

Source code formatting can introduce an error in the generated C++ code. If you specify the code beautification option (`-z`) when you compile your code and an error is displayed when you compile the generated C++ code, try compiling your code without the `-z` option to ensure that the error is not introduced by the source code formatter.

3.4.4. Installing Streams by using the automated method

The automated installation method enables you to build an installation package that you can use to install the Streams product on a large number of networked resources that share the same configuration.

Before you begin

Install a configuration management tool such as Puppet or Chef.

Procedure

1. Install the Streams main installation package on at least one resource. For instructions, see "Installing Streams by using the interactive or silent method. The main installation package includes all of the product files.
2. For additional Streams resources in the domain, create a domain host installation package file that includes a subset of the product files.

   To generate a package file that runs the installation in silent mode and uses the default installation location, enter the following command:

   ```
   streamtool mkhostpkg -d domain-id --zkconnect host:port
   ```

   #### Notes:

   - The *domain-id* that you specify does not need to exist when you enter the **`streamtool mkhostpkg`** command. If you do not specify a *domain-id*, Streams uses the default domain ID of `StreamsDomain`.
   - The `--zkconnect` option specifies the name of one or more host and port pairs for the configured external ZooKeeper ensemble. If you specify multiple host and port pairs, separate each pair with a comma. This value is the external ZooKeeper connection string. To obtain this value, you can use the **`streamtool getzk`** command.
   - The generated package is located in the current directory. To use a different directory, specify the `--pkg-dir` *`pathname`* option on the command.
   - By default, the domain controller service is set up as a registered Linux system service when you install the domain host installation package. To run the controller as an unregistered service, specify the --unregistered option on the command. For more information, see Options for setting up the domain controller service.

- The **mkhostpkg** command generates an uncompressed tar file. The contents of the files in the installation tar file are compressed. The installation package file includes a response file (`responsefile.properties`) that is used to perform the installation. By default, the **mkhostpkg** command creates a domain host installation package, which includes a subset of the product files. For example, this package does not contain the toolkits.

3. **Optional:** Customize the `responsefile.properties` file that was created by the **mkhostpkg** command.

   a. Extract the contents of the installation package.

   b. Use an editor of your choice to customize the response file.

4. Use a configuration management tool, such as Puppet or Chef, to install Streams on each resource. The configuration management tool performs the following operations on the resource:

   - Copies and extracts the contents of the installation package file that you created in Step 2 by using the **streamtool mkhostpkg** command.
   - Optionally runs the Streams `dependency_checker.sh` script. This script must run from the directory where the extracted files are located.
   - As user root installs Streams on the resource by running the streamsdomainhostsetup.sh script. This script must run from the directory where the extracted files are located.

Notes:

- If the installation is successful, Streams creates the *product- installation-root-directory*`/7.1.0.0/.success` file. The creation of this file can be used as a repeatable test in the configuration management tool to determine if the installation is successful.
- The response file contains a property to set the language, country, and character encoding. By default, the value of this property is the language, country, and character encoding of the system on which the installation package was created. This property must have the character encoding set to UTF-8. For example a valid value is **en_US.UTF-8**.

What to do next

Create an enterprise domain. For instructions, see Setting up an enterprise domain.

Related concepts:

"Streams installation packages"

Use the *main installation package* to install all of the product files on the resources in your Streams domain. To reduce the product size requirement on resources, the *domain host installation package* and *resource installation package* contain only a subset of the product files. For example, these installation packages do not include the toolkits.

Related reference:

Streamtool commands

This reference section provides a description of each **`streamtool`** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

3.5. Reviewing the license agreement for Streams

To review the Streams license information, use the interactive installation graphical user interface or a browser.

Procedure

- To access the product license information by using the interactive installation graphical user interface:

  1. Run the Streams self-extracting installation binary file by entering the following commands from an xterm window:

     ```
     cd product-installation-package-directory/StreamsInstallFiles
     ./StreamsSetup.bin
     ```

  2. Continue through the interactive installation panels until the license information is displayed.
  3. Read the license information.
  4. Cancel the installation.

Related tasks:

Tracking license usage

Chapter 4. Installing Streams Studio

Use these procedures to install Streams Studio on a Linux or Microsoft Windows system.

Important

- To update Streams Studio Version 7.1.0 after installing a Version 7.1.0 fix pack or interim fix, use the procedure in "Updating Streams Studio Version 7.1.0".
- To update Streams Studio Version 4.3.x, or later to Version 7.1.0, use the procedure in "Updating Streams Studio to Version 7.1.0".

Before installing Streams Studio, review the software prerequisites.

4.1. Software prerequisites for Streams Studio

Before you install Streams Studio, ensure that you satisfy the prerequisite software requirements.

Table 4.1. Required software for Streams Studio

Required software	Required version	Additional information
Streams	Version 7.1.0	For instructions, see Chapter 3, "Installing the Streams product".
X Window System	Version that is compatible with your operating system.	To verify that an X Window System is installed, see your operating system documentation.
Eclipse SDK	Streams Studio supports Eclipse SDK Version 4.5.2.	Streams Studio has been tested with Version 4.5.2. This version is included in the Streams installation package and is installed when you install Streams Studio.
GTK	GTK+ 2 and its dependencies.	This version is required by the Standard Widget Toolkit (SWT) that is included in the Eclipse SDK. GTK+ 3 is not supported.
Java development environment	For application development, Streams supports the IBM SDK Java Technology Edition Version 8 and the Java SE Development Kit 8 from Oracle. The IBM SDK is installed with Streams Studio.	For additional information about using Java with the product, see "Java requirements, options, and settings for Streams".
Microsoft Windows operating system	If you install Streams Studio on a Windows system for remote development, the required	For installation instructions, see "Installing Streams Studio on a local Microsoft Windows system".

Required software	Required version	Additional information
	operating system version is Windows 10.	

4.2. Installing Streams Studio for local or remote development

You can install Streams Studio on the Linux system where you install Streams, or on a local Linux or Microsoft Windows system that accesses a remote Linux system where Streams is installed.

Before you begin

Install Streams. For instructions, see Chapter 3, “Installing the Streams product”.

4.2.1. Installing Streams Studio for local development

Use this procedure to install Streams Studio on the same Linux system as the Streams product.

Procedure

Root user or Streams installation owner:

1. Download the Streams Studio package from https://21cs.com/streams to the Linux system where Streams is installed.
2. Extract the contents of the Streams Studio package to the following location:

   streams-installation-root-directory`/7.1.0.0/etc`

   This will create the `eclipse`, `rseserver`, and `StreamsStudio` subdirectories.
3. Change the ownership of the three new directories and their contents to the Streams installation owner and group.
4. Make the following file available to all users who will run Streams Studio on the Linux system where Streams is installed:

   streams-installation-root-directory`/7.1.0.0/etc/StreamsStudio/StreamsStudio.tar.gz`

User who will run Streams Studio on the Linux system where Streams is installed:

1. Extract the contents of StreamsStudio.tar.gz to your Streams Studio installation directory on the Linux system where Streams is installed.
2. Enter the following command to start Streams Studio:

   ***studio-installation-directory*`/StreamsStudio/streamsStudio -clean`**

4.2.2. Installing Streams Studio for remote development

Use this procedure to install Streams Studio Version 7.1.0 on a local Linux or Microsoft Windows system.

Streams Studio uses the client-server architecture to provide a seamless remote development environment. To use remote development, you install Streams Studio on a local Linux or Windows system that acts as a client and install Streams on a remote Linux system that acts as a server. After you establish a connection with the remote Linux system, you can use Streams Studio on your local system to develop streams processing applications remotely.

The Streams runtime environment, Streams Processing Language (SPL) compiler, and SPL toolkits are located on the remote Linux system. Streams Studio uses Remote System Explorer (RSE) to provide connectivity to the remote Linux system for building stream processing applications. RSE is included in the Streams Studio installation.

4.2.2.1. Installing Streams Studio on a local Linux system

Use this procedure to install Streams Studio on a local Linux system that will be used for remote development.

Before you begin

Ensure that your local Linux system satisfies the following requirements:

- Linux operating system version that is supported by Streams. For more information, see "Supported operating system versions for Streams".
- Streams Studio prerequisites. For more information, see "Software prerequisites for Streams Studio".

Procedure

Root user or Streams installation owner:

1. Download the Streams Studio package from https://21cs.com/streams to the Linux system where Streams is installed.
2. Extract the contents of the Streams Studio package to the following location:

   streams-installation-root-directory`/7.1.0.0/etc`

   This will create the `eclipse`, `rseserver`, and `StreamsStudio` subdirectories.
3. Change the ownership of the three new directories and their contents to the Streams installation owner and group.
4. Make the following file available to all users who will run Streams Studio on a local Linux system:

   streams-installation-root-directory`/7.1.0.0/etc/StreamsStudio/StreamsStudio.tar.gz`

User who will run Streams Studio on a local Linux system:

1. Extract the contents of StreamsStudio.tar.gz to your Streams Studio installation directory on your local Linux system.
2. Enter the following command to start Streams Studio:

```
studio-installation-directory/StreamsStudio/streamsStudio -clean
```

3. Configure a connection to the remote Linux system where Streams is installed. For instructions, see Configuring the connection for remote development with Streams Studio.

4.2.2.2. Installing Streams Studio on a local Microsoft Windows system

Use this procedure to install Streams Studio on a local Microsoft Windows system that will be used for remote development.

Before you begin

Note that Windows 10 (64-bit) is required for remote development.

Important

Specify a short installation file path for Streams Studio, for example, `C:\Streams`. If the Streams Studio installation file path is too long, it causes the file paths of some extracted files to exceed the length that is allowed by Windows. If this occurs, Windows issues an error message that the path is too long.

Procedure

Root user or Streams installation owner:

1. Download the Streams Studio package from https://21cs.com/streams to the Linux system where Streams is installed.

2. Extract the contents of the Streams Studio package to the following location:

   ```
   streams-installation-root-directory/7.1.0.0/etc
   ```

   This will create the `eclipse`, `rseserver`, and `StreamsStudio` subdirectories.

3. Change the ownership of the three new directories and their contents to the Streams installation owner and group.

4. Make the following file available to all users who will run Streams Studio on a local Windows system:

   ```
   streams-installation-root-directory/7.1.0.0/etc/StreamsStudio/
   StreamsStudio-Win.zip
   ```

User who will run Streams Studio on a local Windows system:

1. Extract the contents of StreamsStudio-Win.zip to your Streams Studio installation directory on your Windows system.

2. Enter the following command to start Streams Studio:

   ```
   studio-installation-directory\StreamsStudio\streamsStudio -clean
   ```

3. Configure a connection to the remote Linux system where Streams is installed. For instructions, see Configuring the connection for remote development with Streams Studio.

4.3. Running Streams Studio from a shared installation

You can install Streams Studio in a shared directory location so that multiple users can share the installation.

Procedure

1. Install Streams Studio in a shared location on the file system.
2. If you install any additional software, ensure that you restart Streams Studio with the `-clean` option after each installation.
3. Allow only read permission for the `StreamsStudio` directory and all of its subdirectories.
4. Ensure that the `StreamStudio/configuration` directory has only read permission for all users.
5. When starting Streams Studio, each user must specify the `-configuration` option as shown in the following example:

   ```
   Studio-installation-directory/streamsStudio -clean -configuration user-write-access-directory
   ```

4.4. Uninstalling Streams Studio

Use this procedure to remove the Streams Studio development environment from your system.

Procedure

Use either of the following methods to uninstall Streams Studio:

- Remove the Streams Studio development environment by deleting the directory where you installed it.
- Remove the Streams Studio plug-ins from an existing Eclipse environment.
  1. From the menu br, click **Help** > **About Streams Studio**.
  2. Click **Installation Details** and uninstall the Streams Studio plug-ins.

Chapter 5. Upgrading Streams

Use these procedures to install a new version of Streams when a previous version of the product is already installed.

5.1. Version management and rolling upgrade options for Streams

Starting from Version 7.1, Streams supports version management and rolling upgrade, which provide the following benefits:

- Managed version support enables you to upgrade a domain and its instances independent of each other. When you upgrade a domain, its instances continue to run at their current version. With managed version support, you can upgrade the instances immediately after the domain is upgraded or at a later time. Instances can also be upgraded independent of each other.

- Rolling upgrade support enables you to upgrade a domain or instance to a new version of Streams while the domain or instance is running.

Warning

Information in this section does *not* apply if you are upgrading from an earlier version of Streams.

Upgrade options

When you upgrade Version 7.1, you can perform a standard or rolling upgrade.

- *Standard upgrade*: If a domain or instance is stopped, you can upgrade the domain or instance by simply starting it with the new version.

    1. To start a domain with the new version, you can use the **`streamtool startdomain`** command.

    2. To start an instance with the new version, you can use the Streams Console, Streams Studio, or the **`streamtool startinstance`** command.

    You can use this same procedure to switch back to an earlier installed Streams version.

- *Rolling upgrade*: You can perform an upgrade while the domain or instance is running.

    - To upgrade a domain to the new version, you can use the **`streamtool upgradedomain`** command.

    - To upgrade an instance to the new version, you can use the Streams Console or the **`streamtool upgradeinstance`** command.

Domain and instance version considerations

In some cases, you might want to upgrade a domain without upgrading all of its instances. For example:

- You might have applications that require an earlier version than the version that is running on the domain.
- You might want to run the new version on a test instance and run the earlier version on the production instances for a period of time.

If a domain and its instances are running different versions, all versions must be installed in the same root directory. In addition, the domain must run at the same or a later version than all of the instances.

When a domain is upgraded, its instances continue to run at their current version. By default, when you restart an instance after the domain is upgraded, Streams upgrades the instance to the domain version. You can use the **`instance.startAsVersion`** and **`domain.determineInstanceStartAsVersion`** properties to override the default behavior.

instance.startAsVersion

Indicates the Streams version that is used to start the instance. This version must be installed on the resource where the instance is started, and it must be installed in the same installation root directory that is used to start the domain.

For more information about this property, enter **`streamtool man properties`**. To update this property, use the **`streamtool setproperty`** command.

Notes

- If you use the **`streamtool startinstance`** command to start an instance, you can use the **`--version`** command option to specify the version to run on the instance. This option takes precedence over the **`instance.startAsVersion`** property value.
- If you use the Streams Console or Streams Studio to start an instance, the versions available to start the instance are provided in a selection list. The default selection value is the value of the **`instance.startAsVersion`** property.

---

domain.determineInstanceStartAsVersion

Indicates whether the **`instance.startAsVersion`** property is automatically set when an instance is created or upgraded.

- If the default value of false is used, the **`instance.startAsVersion`** property value is not changed when an instance is created or upgraded.
- If the value is set to true, the version that is set for the **`instance.startAsVersion`** property is the version of the Streams interface that is used to create or upgrade the instance.

For more information about this property, enter `streamtool man domainproperties`. To update this property, use the **`streamtool setdomainproperty`** command.

In a mixed version environment, you must manage the domain by using the Streams version that is running on the domain. For example, if you use the **`streamtool`** command-line interface, you must run commands from the installation directory for the domain version and not a different version that is running on an instance in the domain.

Job and PE version considerations

Let's assume for example, you have a Streams domain with instances and jobs running in a previous Streams release, such as Version 4.2, and you install Version 7.1 on at least one domain resource:

- You can perform a rolling upgrade on your domain and instances without stopping and restarting all jobs and PEs.
- All PEs continue to run in a healthy state while the instance is upgrading and after the upgrade is completed.
- All existing PEs that were running before the upgrade is completed continue to run with their original release version, in this example, Streams Version 4.3.x or 4.3 or later.
- Any new job that you submit to the upgraded instance runs with the release version of the running instance, in this example, Streams Version 7.1.
- You can use the **`streamtool lsjobs --long`** command option to track the release version used by each running job or PE. The **ProductVersion** column in the **`streamtool lsjobs`** command result indicates the release version of running jobs and PEs.

5.2. Upgrading to Streams Version 7.1.0

Use these procedures to install Version 7.1.0 when Versions 4.0 or later is already installed. You must perform a new product installation by using the procedure in Installing the Streams product.

5.2.1. Upgrading to Version 7.1.0 when the default Streams resource manager is used

If you are using resources that are managed by the default Streams resource manager only, use these procedures to install Streams Version 7.1.0.

5.2.1.1. Upgrading when Version 4.3.x is installed

If Version 4.3.x is already installed and you are using resources that are managed by the default Streams resource manager only, use this procedure to install Streams Version 7.1.0.

This procedure installs Streams on resources that are managed by the default Streams resource manager. These resources have the main installation package or the domain host installation package installed.

Before you begin

Ensure that you satisfy the following requirements:

- To upgrade the product, you must be the installation owner or have root authority. If you are installing the product as a root user, you must select the same installation owner as the previous version when prompted.
- You must complete this procedure on all resources that have the main installation package installed. This package includes all of the product files. Additional resources with the domain host installation package installed are automatically updated. Streams provisions all updates to these additional resources in the domain.

Important: For the upgrade and automatic resource updates to work correctly, you must install the new version in the same root directory as the previous version.

About this task

When you install the product in the same root directory as the previous version, Streams creates a new version directory for the installation. The version directory for the previous installation is not removed.

If an X Window System is installed, the default mode for upgrading Streams is interactive GUI. Otherwise, the default mode is interactive console.

Procedure

1. Download the Version 7.1.0 installation package.
2. Install Streams on all resources that have the main installation package installed by using one of the following procedures:
   - Interactive GUI installation
   - Interactive console installation
   - Silent installation

   If you are using the automated installation method to upgrade Streams, install the product on each resource by using a configuration management tool such as Puppet or Chef.
3. After installing Streams, complete the following steps:
   a. Configure the Streams environment variables for the new version by entering the following command:

      ```
      source product-installation-root-directory/product-version/bin/
      streamsprofile.sh
      ```

      Note

      If you added the **source** command for the previous version to the `.bashrc` or `/etc/profile.d` file, update the version directory in those files.
4. For better **streamtool** performance, clear the **streamtool** Java cache. For more information, see Java cache for Streams streamtool command operations.
5. Stop all applications, instances, and domains. For more details, refer to Section 6.1
6. Upgrade the domain by using the standard upgrade procedure:
   - If the domain is stopped, you can upgrade the domain by simply starting it with the new version. To start the domain with the new version, you can enter the following command:

     ```
     streamtool startdomain -d domain-id
     ```
   - If the domain is *not* stopped, you must stop all instances and the domain *before* running the standard upgrade procedure.

   Streams completes the following operations:

   a. Provisions all updates to additional resources in the domain that have the domain host installation package installed.

b. Starts the domain controller service.

c. Starts the domain services with the new version.

7. Upgrade the instances in the domain by using the standard upgrade procedure. To start the instance with the new version, you can use the Streams Console, Streams Studio, or the following command:

   ```
   streamtool startinstance -d domain-id -i instance-id
   ```

8. Verify that the upgrade is successful by entering the following commands:

   ```
   streamtool getdomainstate --long
   ```

   If the installation is successful, the new version is the active version. In the version column of the command output, an asterisk (*) indicates which version is active.

   ```
   streamtool lsinstance --long
   ```

   If the upgrade is successful, the new version is listed in the Version column.

9. Resubmit all applications by using entering the following command:

   ```
   streamtool submitjob -d domain-id -i instance-id application-bundle-name.sab
   ```

Related concepts:

"Streams installation packages"

Use the *main installation package* to install all of the product files on the resources in your Streams domain. To reduce the product size requirement on resources, the *domain host installation package* and *resource installation package* contain only a subset of the product files. For example, these installation packages do not include the toolkits.

Related tasks:

"Configuring the Streams environment by running streamsprofile.sh"

Before you can use Streams, you must configure the product environment variables by running the **streamsprofile.sh** script that is included in the installation directory. This script is not automatically run by the installation program.

Related reference:

"Migration guidelines for Version 4.3 releases"

If you are upgrading Streams and a Version 4.3.x or 4.3 or later release is already installed, review these migration guidelines.

Resource managers

Streamtool commands

This reference section provides a description of each **streamtool** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man command-name`.

5.2.2. Upgrading to Version 7.1.0 when an external resource manager is used

If you are using resources that are managed by a supported external resource manager, use these procedures to install Streams Version 7.1.0. Any resources that are managed by the default Streams resource manager are also upgraded.

These procedures install Streams on the following resources:

- Resources that are managed by a supported external resource manager or a user-defined resource manager. These resources have the main installation package or resource installation package installed.
- Any resources that are managed by the default Streams resource manager. These resources have the main installation package or domain host installation package installed.

5.2.2.1. Upgrading when a user-defined resource manager is used

If you are using resources that are managed by a user-defined resource manager, use these procedures to install Streams Version 7.1.0.

Any resources that are managed by the default Streams resource manager are also upgraded.

Upgrading when Version 4.3.x is installed

If Version 4.3.x is already installed and you are using resources that are managed by a user-defined resource manager, use this procedure to install Streams Version 7.1.0.

Before you begin

Ensure that you satisfy the following requirements:

- To upgrade the product, you must be the installation owner or have root authority. If you are installing the product as a root user, you must select the same installation owner as the previous version when prompted.
- You must complete this procedure on all resources that have the main installation package installed. This package includes all of the product files. Additional resources with the resource installation package or domain host installation package installed are automatically updated. Streams provisions all updates to these additional resources in the domain.

> **Important**
>
> For the upgrade and automatic resource updates to work correctly, you must install the new version in the same root directory as the previous version.

About this task

When you install the product in the same root directory as the previous version, Streams creates a new version directory for the installation. The version directory for the previous installation is not removed.

If an X Window System is installed, the default mode for upgrading Streams is interactive GUI. Otherwise, the default mode is interactive console.

Procedure

1. Download the installation package.
2. Install Streams on all resources that have the main installation package installed by using one of the following procedures:
   - Interactive GUI installation
   - Interactive console installation
   - Silent installation
3. After installing Streams, configure the Streams environment variables for the new version by entering the following command:

   ```
   source product-installation-root-directory/product-version/bin/
   streamsprofile.sh
   ```

   Note

   If you added the *source* command for the previous version to the `.bashrc` or `/etc/profile.d` file, update the version directory in those files.

4. Install the resource installation package on the externally managed resources.
   a. Stop the user-defined resource manager.
   b. Create and deploy a resource installation package.
   c. Start the user-defined resource manager.
5. Stop all applications, instances, and domains. For more details, refer to Section 6.1
6. Upgrade the domain by using the standard upgrade procedure:
   - If the domain is stopped, you can upgrade the domain by simply starting it with the new version. To start the domain with the new version, you can enter the following command:

     ```
     streamtool startdomain -d domain-id
     ```

   - If the domain is not stopped, you must stop all instances and the domain *before* running the standard upgrade procedure.

   Streams completes the following operations:
   a. Provisions all updates to additional resources in the domain that have the domain host installation package installed.
   b. Starts the domain controller service.
   c. Starts the domain services with the new version.
7. Upgrade the instances in the domain by using the standard upgrade procedure.

   To start the instance with the new version, you can use the Streams Console, Streams Studio, or the following command:

```
streamtool startinstance -d domain-id -i instance-id
```

8. Verify that the upgrade is successful by entering the following commands:

   ```
   streamtool getdomainstate --long
   ```

   If the installation is successful, the new version is the active version. In the version column of the command output, an asterisk (*) indicates which version is active.

   ```
   streamtool lsinstance --long
   ```

   If the upgrade is successful, the new version is listed in the Version column.

9. Resubmit all applications by entering the following command:

   ```
   streamtool submitjob -d domain-id -i instance-id application-bundle-name.sab
   ```

Related concepts:

"Streams installation packages"

Use the *main installation package* to install all of the product files on the resources in your Streams domain. To reduce the product size requirement on resources, the *domain host installation package* and *resource installation package* contain only a subset of the product files. For example, these installation packages do not include the toolkits.

Related tasks:

"Configuring the Streams environment by running streamsprofile.sh"

Before you can use Streams, you must configure the product environment variables by running the **`streamsprofile.sh`** script that is included in the installation directory. This script is not automatically run by the installation program.

Related reference:

"Migration guidelines for Version 4.3.x releases"

If you are upgrading Streams and a Version 4.3.x release is already installed, review these migration guidelines.

Resource managers

Streamtool commands

This reference section provides a description of each **`streamtool`** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man command-name`.

5.2.3. Migration guidelines for Streams Version 7.1.0

If you are upgrading to Version 7.1.0 and a previous version of Streams is already installed, review these migration guidelines before you use the new version.

5.2.3.1. Migration guidelines for Version 4.3.x releases

If you are upgrading Streams and a Version 4.3.x release is already installed, review these migration guidelines.

Updating Streams Studio to Version 7.1.0

Use this procedure to update an existing Streams Studio Version 4.3.x environment to Version 7.1.0.

Before you begin

Upgrade the Streams product.

To reuse your existing Streams Studio installation, you must use the following procedure to update Streams Studio. Otherwise, your existing Streams Studio directory and all its contents are removed. This includes the Streams Studio workspaces that reside under the Streams Studio installation directory, customized tools, and other customizations of the Streams Studio development environment.

Procedure

Root user or Streams installation owner:

1. Download the Streams Studio package from https://21cs.com/streams to the Linux system where Streams is installed.
2. Extract the contents of the Streams Studio package to the following location:

   streams-installation-root-directory`/7.1.0.0/etc`

   This will create the `eclipse`, `rseserver`, and `StreamsStudio` subdirectories.
3. Change the ownership of the three new directories and their contents to the Streams installation owner and group.

Streams Studio user:

1. The Streams Update Site package is installed in the following directory on the Linux system where Streams is installed:

   streams-installation-root-directory`/7.1.0.0/etc/eclipse`

   If Streams Studio is installed for remote development on your local Linux or Windows system, download the Studio Update Site files from the directory mentioned above.
2. Start Streams Studio with a new workspace.
3. In the **Edit Install Location** dialog, click **Cancel**.
4. In the **Missing Streams Install** dialog, click **OK**.
5. From the top navigation menu, click **Window** > **Preferences**.
6. Add the Streams Studio Update Site by following these steps:
   a. In the **Preferences** window, expand **Install/Update**.

b. Click **Available Software Sites**.

c. Click **Add**.

d. In the **Name** field, enter `Streams Studio Update Site`.

e. Click **Local**.

f. Navigate to the eclipse directory containing the Studio Update Site files. For remote development, this is the directory where you copied the Studio Update Site files in Step 1. For local development, this is the following directory:

streams-installation-root-directory`/7.1.0.0/etc/eclipse`

g. Click **OK** to close all open dialogs.

7. Click **Help** > **Check for Updates** to install the Streams Studio updates. The update program will guide you through the update process.

Warning

At some point during the update, you will see a dialog asking you to restart Streams Studio. If you are performing the update on a Linux system, *do not* interact with this dialog at this stage and proceed to the next step.

---

8. **If you are on Linux:** Update the streamsStudio.ini file, which is located in the root directory of the Streams Studio installation. Add the following two lines before the `-vmargs` line:

```
--launcher.GTK_version
2
```

9. Restart Streams Studio.

10. Configure the Streams install location by following these steps:

a. For local development, enter the install directory of the updated Streams product in the **Edit Install Location** dialog.

b. For remote development, follow the instructions in Configuring Streams Studio for remote development.

11. When prompted, provide the Streams Domain connection information.

12. From the top navigation menu, go to **File** > **Import** > **Streams Studio** > **SPL Project**, then click **Next**.

13. Navigate to the location of your previous workspace directory and click **OK**.

14. Select the projects to import and click **Finish**.

Related tasks:

Importing SPL projects

Use this procedure to import an SPL project into Streams Studio.

5.3. Upgrading Streams Version 7.1.0

Use these procedures to install a Streams Version 7.1.0 fix pack or interim fix when Version 7.1.0 is already installed. If an earlier version is installed, use the procedures in Upgrading to Streams Version 7.1.0.

Streams product updates include a complete version of the product. When you install a Version 7.1.0 fix pack or interim fix, the installation procedure installs Version 7.1.0 and all the product updates for that version.

After installing the fix pack or interim fix, you can update Streams Studio.

5.3.1. Installing a fix pack or interim fix when the default Streams resource manager is used

If you are using resources that are managed by the default Streams resource manager only, use this procedure to install a Streams Version 7.1.0 fix pack or interim fix when Version 7.1.0 is already installed.

This procedure installs the fix pack or interim fix on resources that are managed by the default Streams resource manager. These resources have the main installation package or the domain host installation package installed.

This procedures includes version management and rolling upgrade options. Managed version support enables you to upgrade a domain and its instances independent of each other. Rolling upgrade support enables you to upgrade a domain or instance while it is running. For more information, see Version management and rolling upgrade options.

Before you begin

Ensure that you satisfy the following requirements:

- To install a Streams fix pack or interim fix, you must be the installation owner or have root authority. If you are installing the fix as a root user, you must select the same installation owner as the previous version when prompted.
- You must complete this procedure on all resources that have the main installation package installed. This package includes all of the product files. Additional resources with the domain host installation package installed are automatically updated. Streams provisions all updates to these additional resources in the domain.

Important

For the upgrade and automatic resource updates to work correctly, you must install the new version in the same root directory as the previous version.

About this task

When you install the fix pack or interim fix in the same root directory as the previous version, Streams creates a new version directory for the installation. The version directory for the previous installation is not removed.

If an X Window System is installed, the default mode for installing a fix pack or interim fix is interactive GUI. Otherwise, the default mode is interactive console.

Procedure

1. Download the fix pack or interim fix.
2. Install the fix pack or interim fix on all resources that have the main installation package installed by using one of the following procedures:
   - Interactive GUI installation
   - Interactive console installation
   - Silent installation

   If you are using the automated installation method to upgrade Streams, install the fix pack or interim fix on each resource by using a configuration management tool such as Puppet or Chef.
3. After installing Streams, complete the following steps:
   a. Configure the Streams environment variables for the new version by entering the following command:

      ```
      source product-installation-root-directory/product-version/bin/
      streamsprofile.sh
      ```

      Note

      If you added the **`source`** command for the previous version to the `.bashrc` or `/etc/profile.d` file, update the version directory in those files.

   b. Optional: Set the **`instance.startAsVersion`** and **`domain.determineInstanceStartAsVersion`** version management properties. For more information about these properties, see Version management and rolling upgrade options.
4. For better *streamtool* performance, clear the *streamtool* Java cache. For more information, see Java cache for Streams streamtool command operations.
5. Upgrade the domain by using the standard or rolling upgrade procedure.
   - Standard upgrade: If the domain is stopped, you can upgrade the domain by simply starting it with the new version. To start the domain with the new version, you can enter the following command:

     ```
     streamtool startdomain -d domain-id
     ```

   - Rolling upgrade: If the domain is running, enter the following command:

     ```
     streamtool upgradedomain -d domain-id
     ```

   Streams completes the following operations:

   a. Provisions all updates to additional resources in the domain that have the domain host installation package installed.

   b. Restarts the domain controller service.

   c. Restarts the domain services with the new version.

Instances in the domain continue to run using the installation of the previous version.

6. Upgrade the instances in the domain by using the standard or rolling upgrade procedure.

   You can upgrade the instances immediately after the domain is upgraded or at a later time. If upgraded later, you can either upgrade the instances at the same time or different times, depending on your requirements.

   - Standard upgrade: If the instance is stopped, you can upgrade the instance by simply starting it with the new version. To start the instance with the new version, you can use the Streams Console, Streams Studio, or the following command:

     ```
     streamtool startinstance -d domain-id -i instance-id
     ```

   - Rolling upgrade: If the instance is running, you can use the Streams Console upgrade option for the instance or enter the following command:

     ```
     streamtool upgradeinstance -d domain-id -i instance-id
     ```

     You can specify multiple instance IDs on this command, or specify the `-\-all` option to upgrade all instances in the domain.

7. Verify that the installation of the fix pack or interim fix is successful by entering the following commands:

   ```
   streamtool getdomainstate --long
   ```

   If the installation is successful, the fix pack or interim fix version is the active version. In the version column of the command output, an asterisk (*) indicates which version is active.

   ```
   streamtool lsinstance --long
   ```

   If the upgrade is successful, the fix version is listed in the Version column.

Related concepts:

"Streams installation packages"

Use the *main installation package* to install all of the product files on the resources in your Streams domain. To reduce the product size requirement on resources, the *domain host installation package* and *resource installation package* contain only a subset of the product files. For example, these installation packages do not include the toolkits.

Related tasks:

"Configuring the Streams environment by running streamsprofile.sh"

Before you can use Streams, you must configure the product environment variables by running the *streamsprofile.sh* script that is included in the installation directory. This script is not automatically run by the installation program.

Related reference:

Resource managers

Streamtool commands

This reference section provides a description of each *streamtool* command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

5.3.2. Installing a fix pack or interim fix when an external resource manager is used

If you are using resources that are managed by a supported external resource manager, use these procedures to install a Streams Version 7.1.0 fix pack or interim fix when Version 7.1.0 is already installed. Any resources that are managed by the default Streams resource manager are also upgraded.

This procedure installs the fix pack or interim fix on the following resources:

- Resources that are managed by a supported external resource manager or a user-defined resource manager. These resources have the main installation package or resource installation package installed.
- Any resources that are managed by the default Streams resource manager. These resources have the main installation package or domain host installation package installed.

These procedures include version management and rolling upgrade options. Managed version support enables you to upgrade a domain and its instances independent of each other. Rolling upgrade support enables you to upgrade a domain or instance while it is running.

5.3.2.1. Installing a fix pack or interim fix when a user-defined resource manager is used

If you are using a user-defined resource manager to manage resources, use this procedure to install a Streams Version 7.1.0 fix pack or interim fix.

Before you begin

Ensure that you satisfy the following requirements:

- To install a Streams fix pack or interim fix, you must be the installation owner or have root authority. If you are installing the fix as a root user, you must select the same installation owner as the previous version when prompted.
- You must complete this procedure on all resources that have the main installation package installed. This package includes all of the product files. Additional resources with the resource installation package or domain host installation package installed are automatically updated. Streams provisions all updates to these additional resources in the domain.

Important

For the upgrade and automatic resource updates to work correctly, you must install the new version in the same root directory as the previous version.

About this task

When you install the fix pack or interim fix in the same root directory as the previous version, Streams creates a new version directory for the installation. The version directory for the previous installation is not removed.

If an X Window System is installed, the default mode for installing a fix pack or interim fix is interactive GUI. Otherwise, the default mode is interactive console.

Procedure

1. Download the fix pack or interim fix.
2. Install the fix pack or interim fix on all resources that have the main installation package installed by using one of the following procedures:
   - Interactive GUI installation
   - Interactive console installation
   - Silent installation
3. After installing Streams, complete the following steps:
   a. Configure the Streams environment variables for the new version by entering the following command:

      ```
      source product-installation-root-directory/product-version/bin/
      streamsprofile.sh
      ```

      > **Note**
      >
      > If you added the **`source`** command for the previous version to the `.bashrc` or `/etc/profile.d` file, update the version directory in those files.

   b. Optional: Set the **`instance.startAsVersion`** and **`domain.determineInstanceStartAsVersion`** version management properties.
4. For better *streamtool* performance, clear the *streamtool* Java cache. For more information, see Java cache for Streams streamtool command operations.
5. Install the resource installation package on the externally managed resources.
   a. Stop the user-defined resource manager.
   b. Create and deploy a resource installation package.
   c. Start the user-defined resource manager.
6. Upgrade the domain by using the standard or rolling upgrade procedure.
   - Standard upgrade: If the domain is stopped, you can upgrade the domain by simply starting it with the new version. To start the domain with the new version, you can enter the following command:

     ```
     streamtool startdomain -d domain-id
     ```

   - Rolling upgrade: If the domain is running, you can enter the following command:

     ```
     streamtool upgradedomain -d domain-id
     ```

   Streams completes the following operations:

a. Provisions all updates to additional resources in the domain that have the domain host installation package installed.

b. Restarts the domain controller service.

c. Restarts the domain services with the new version.

Instances in the domain continue to run using the installation of the previous version.

7. Upgrade the instances in the domain by using the standard or rolling upgrade procedure.

   You can upgrade the instances immediately after the domain is upgraded or at a later time. If upgraded later, you can either upgrade the instances at the same time or different times, depending on your requirements.

   - Standard upgrade: If the instance is stopped, you can upgrade the instance by simply starting it with the new version. To start the instance with the new version, you can use the Streams Console, Streams Studio, or the following command:

     ```
     streamtool startinstance -d domain-id -i instance-id
     ```

   - Rolling upgrade: If the instance is running, you can use the Streams Console upgrade option for the instance or enter the following command:

     ```
     streamtool upgradeinstance -d domain-id -i instance-id
     ```

     You can specify multiple instance IDs on this command, or specify the `--all` option to upgrade all instances in the domain.

8. Verify that the installation of the fix pack or interim fix is successful by entering the following commands:

   ```
   streamtool getdomainstate --long
   ```

   If the installation is successful, the fix pack or interim fix version is the active version. In the version column of the command output, an asterisk (*) indicates which version is active.

   ```
   streamtool lsinstance --long
   ```

   If the upgrade is successful, the fix version is listed in the Version column.

Related concepts:

“Streams installation packages”

Use the *main installation package* to install all of the product files on the resources in your Streams domain. To reduce the product size requirement on resources, the *domain host installation package* and *resource installation package* contain only a subset of the product files. For example, these installation packages do not include the toolkits.

Related tasks:

“Configuring the Streams environment by running streamsprofile.sh”

Before you can use Streams, you must configure the product environment variables by running the *streamsprofile.sh* script that is included in the installation directory. This script is not automatically run by the installation program.

Related reference:

Resource managers

Streamtool commands

This reference section provides a description of each **streamtool** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man command-name`.

5.3.3. Updating Streams Studio Version 7.1.0

Use this procedure to update an existing Streams Studio Version 7.1.0 environment after you install a Streams fix pack or interim fix.

Before you begin

Install the Streams Version 7.1.0 fix pack or interim fix.

To reuse your existing Streams Studio installation, you must use the following procedure to update Streams Studio. Otherwise, your existing Streams Studio directory and all its contents are removed. This includes the Streams Studio workspaces that reside under the Streams Studio installation directory, customized tools, and other customizations of the Streams Studio development environment.

Procedure

1. Download the Streams Studio update package from the product website and extract the contents into a Streams Studio update directory on the system where Streams Studio runs.
2. In Streams Studio, click **Window** > **Preferences**.
3. Add the Streams Studio update site.
   a. In the Preferences window, expand **Install/Update**.
   b. Click **Available Software Sites**.
   c. Click **Add**.
   d. In the **Name** field, enter Streams Studio Update Site.
   e. Click **Local** to find the **Location** on your local system.
   f. Select the update directory where you extracted the update files in Step 1.
   g. Click **OK**. The directory is added to the **Location** field.
   h. Click **OK** again to return to the Available Software Sites list. The Streams Studio Update Site is added to the list.
4. Click **OK** to close the Preferences window.
5. Click **Help** > **Check for Updates** to install the Streams Studio updates. The update program guides you through the update process.

Related tasks:

Importing SPL projects

Use this procedure to import an SPL project into Streams Studio.

Chapter 6. Uninstalling Streams

Use this procedure to remove the Streams product from your system.

Before you begin

To uninstall Streams, you must be the installation owner or have root authority.

Stop the applications, instances, and domains that are using the version of the product that you are uninstalling. For more information, see Stopping Streams applications, instances, and domains.

Enter the following command as user root to unregister the resource from the domain and stop the domain controller service on the resource:

```
streamtool unregisterdomainhost -d domain-id --zkconnect host:port
```

The `--zkconnect` option specifies the name of one or more host and port pairs for the configured external ZooKeeper ensemble. If you specify multiple host and port pairs, separate each pair with a comma. This value is the external ZooKeeper connection string. To obtain this value, you can use the **`streamtool getzk`** command.

About this task

The default mode for uninstalling Streams is interactive console mode.

Procedure

1. Interactive console method
   a. Enter the following commands:

      ```
      cd product-installation-root-directory/7.1.0.0/uninstall
      ./uninstallstreams
      ```

   b. Follow the instructions in the prompts.

   c. After you review the summary panel, respond *Y* to uninstall the product.

2. Silent method
   a. To start the program, enter the following commands:

      ```
      cd product-installation-root-directory/7.1.0.0/uninstall
      ./uninstallstreams –i silent
      ```

Results

Streams removes all the files that were installed by the Streams installation program from your system.

What to do next

If you do not plan to reinstall Streams, have no other versions of the product installed, and no longer need the configuration data for the product, remove the *home-directory*`/.streams` directory. To remove this directory, enter the following command:

```
rm -rf home-directory/.streams
```

Related reference:streamtool unregisterdomainhost command

This description provides information about command syntax and options.

6.1. Stopping Streams applications, instances, and domains

Before you uninstall Streams, stop the applications, instances, and domains that are using the version of the product that you are uninstalling. If the applications, instances, and domains that are using the installation are not stopped, some files for the previous installation might not be removed. The presence of old files can result in unpredictable application behavior.

Procedure

1. Stop all applications that are using the installation. This includes the Streams Console, Streams Studio, and any other application that is using the current installation.
2. Stop all instances that are using the installation by using the **`streamtool stopinstance`** command.
3. Stop all domains that are using the installation by using the **`streamtool stopdomain`** command.

Related reference: Streamtool commands

This reference section provides a description of each **`streamtool`** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

Part III. Configuring Streams

Chapter 7. Setting up a Streams basic domain on a single resource

A *basic domain* has a single Streams resource and user. This type of domain is typically used for test or development environments.

Streams uses the Pluggable Authentication Module (PAM) authentication service for user authentication, and Apache ZooKeeper for managing and storing configuration information. By default, Streams uses an embedded version of ZooKeeper, which is included in the product installation.

7.1. Creating a Streams basic domain and instance

You can create a basic domain and instance by using the Streams graphical user interfaces or the *streamtool* command-line interface.

A basic domain is typically used for test or development environments. Use an enterprise domain for a production environment.

Before you begin

- Ensure that the value of the STREAMS_ZKCONNECT environment variable is unset. This environment variable is used to set the connection string for an external ZooKeeper server. If the value of this environment variable is not unset, Streams **`streamtool`** commands might not use embedded ZooKeeper, which can cause incorrect results.
- If your home directory is on a shared file system, you can only use embedded ZooKeeper on one resource at a time. Streams starts embedded ZooKeeper when you start the domain. For example, if user1 starts the domain on resource A and the same user tries to start the same domain on resource B, an error occurs.
- The **`streams.zookeeper.quorum`** bootstrap property indicates the host name of the embedded ZooKeeper server. Only a local host name is supported. When **`streamtool`** is first started, this property is populated with a local host name and persisted in the home directory of the user. If your home directory is on a shared file system and you want to start embedded ZooKeeper on a different resource, update the **`streams.zookeeper.quorum`** property with the new host name by using the **`streamtool setbootproperty`** command. Otherwise, an error message similar to the following example is issued:
  `CDISA5252E Invalid value for bootstrap property 'streams.zookeeper.quorum'`

Procedure

- Use this procedure to create a basic domain and instance by using the Streams graphical user interfaces.

  Attention: To use the Streams graphical user interfaces, you must have an X Window System installed. An alternative is to use the **`streamtool`** command-line interface procedure.

- Use this procedure to create a basic domain and instance by using the Streams **`streamtool`** command-line interface.

1. To configure your local shell environment for Streams, enter the following command:

   ```
   source product-installation-root-directory/7.1.0.0/bin/
   streamsprofile.sh
   ```

2. To complete this procedure in the interactive **streamtool** interface, enter the following command:

   ```
   streamtool
   ```

   Note

   - When prompted, press Enter to use embedded ZooKeeper.
   - Using the interactive *streamtool* interface saves you time. Streams caches some command options and information so that you do not have to reenter them. Also, you do not have to specify streamtool before each command. To exit the interactive *streamtool* interface, enter exit or quit.

   ---

3. To create a basic domain, enter the following command:

   ```
   mkdomain -d domain-id
   ```

   For example:

   ```
   mkdomain -d StreamsDomain
   ```

4. To create public and private keys for Streams, enter the following command:

   ```
   genkey
   ```

   Generating public and private keys eliminates the need for you to enter a password when you perform Streams tasks that require authentication.

   The **streamtool genkey** command generates the following files:

   - Private key: *user-id*_priv.pem
   - Public key: *user-id*.pem

   Streams stores the private key in the *user-home-directory*/.streams/ key/*domain-id* directory. The public key is stored in ZooKeeper.

5. To start the domain, enter the following command:

   ```
   startdomain
   ```

   Tip: If the domain fails to start because a port is in use, you can change the port number by using the **streamtool setdomainproperty** command. For example, if the domain fails to start because the management API service and web management service ports are in use, you can enter the following command. If you specify a value of 0 for the port number, Streams selects an available port.

   ```
   setdomainproperty jmx.port=0 sws.port=0
   ```

Dynamic port allocation is not ideal, however, if you have applications that use the REST API or JMX API or use tools (such as Streams Studio or Streams Console) that access those services.

6. Create one or more instances for running stream processing applications. To create an instance, enter the following command:

   ```
   mkinstance -i instance-id
   ```

   For example:

   ```
   mkinstance -i StreamsInstance
   ```

7. Start the instance by entering the following command:

   ```
   startinstance
   ```

8. To manage and monitor the domain, you can use the Streams Console. The domain must be started to use the console.

   To open the console:

   a. Enter `geturl` to display the URL for the console. For example, if your resource is myhost.mydomain.com and the port number for the console is 8440, this command displays the following URL:

      ```
      https://myhost.mydomain.com:8440/streams/domain/console
      ```

   b. Paste this URL into your browser.

9. To exit the interactive **`streamtool`** interface, enter exit or quit.

Developing streams processing applications

You develop applications to operate on streaming data by using the Streams Processing Language (SPL). The preferred tool to develop applications is Streams Studio, which contains both a text based SPL editor and a drag-and-drop based SPL graphical editor.

Related reference: Streamtool commands

This reference section provides a description of each *streamtool* command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man command-name`.

Sample applications and tutorials

The Commodity Purchasing sample application is a complete end-to-end example of a streams processing application that runs against live data and provides a user interface. See the tutorials for help in developing streams processing applications.

Troubleshooting ZooKeeper problems

Use these solutions to resolve problems that you might encounter with external ZooKeeper or embedded ZooKeeper.

7.2. Changing the port number for embedded ZooKeeper

Use this procedure to change the port number that Streams uses for embedded ZooKeeper. If the port number you specify is not available, embedded ZooKeeper finds an available port when it starts.

Before you begin

Stop the domain by entering the following command:

```
streamtool stopdomain -d domain-id --embeddedzk
```

Procedure

1. To display the embedded ZooKeeper property settings, enter the following command:

   ```
   streamtool getbootproperty --all
   ```

   The **`streams.zookeeper.property.clientPort`** property value is the embedded ZooKeeper port number, as shown in the following example:

   ```
   streams.zkmonitor.checkIntervalMs=50000
   streams.zkmonitor.connTimeoutMs=5000
   streams.zkmonitor.port=21820
   streams.zookeeper.jvmFlags=
   streams.zookeeper.logDir=/home/myuser/.streams/var/embeddedzk
   streams.zookeeper.property.autopurge.purgeInterval=1
   streams.zookeeper.property.autopurge.snapRetainCount=3
   streams.zookeeper.property.clientPort=21810
   streams.zookeeper.property.dataDir=/home/myuser/.streams/var/embeddedzk/
   datadir
   streams.zookeeper.property.initLimit=10
   streams.zookeeper.property.maxClientCnxns=0
   streams.zookeeper.property.maxSessionTimeout=180000
   streams.zookeeper.property.minSessionTimeout=-1
   streams.zookeeper.property.syncLimit=5
   streams.zookeeper.property.tickTime=3000
   streams.zookeeper.quorum=myhost.com
   streams.zookeeper.rootLogger=INFO,CONSOLE,ROLLINGFILE
   ```

2. To change the port number, use the **`streamtool setbootproperty`** command. For example, to change the port number to 21811, enter the following command:

   ```
   streamtool setbootproperty streams.zookeeper.property.clientPort=21811
   ```

   Note

   Do not use a delimiter, such as a comma, when you specify the port number. For example, specifying 21,811 results in an error.

   ---

3. Start the domain by entering the following command:

   ```
   streamtool startdomain -d domain-id --embeddedzk
   ```

Chapter 8. Setting up a Streams enterprise domain on multiple resources

An *enterprise domain* can have multiple resources and users. This type of domain is typically used for production environments. You can configure high availability to ensure that Streams can continue to run even if resources fail or are not available.

For high availability, you need to set up and configure an Apache ZooKeeper server for managing and storing configuration information. The method that you use to authenticate users in the domain must also be highly available.

The type of enterprise domain that you create depends on your authentication mechanism. The following options are available as default authentication methods:

- Enterprise PAM domain: For high availability, use a Pluggable Authentication Module (PAM) with the LDAP backend. PAM must be set up and configured before you create the domain. The LDAP server must be accessible from the resource that is running the authentication and authorization services. There are other options that might not be highly available, such as PAM with a UNIX backend. For this option, the Streams users must be defined on this system.
- Enterprise LDAP domain: For high availability, use a Lightweight Directory Access Protocol (LDAP) server. LDAP must be set up and configured before you create the domain.

You can obtain resources from the default Streams resource manager or an external resource manager. Streams also supports user-defined external resource managers. Both Streams resources and externally managed resources can be used together in a domain.

8.1. Prerequisites for a Streams enterprise domain

Before you create an enterprise domain, you must set up an external ZooKeeper server for managing Streams. You can set up either LDAP or PAM to use as the default user authentication method for the domain. To manage resources, you can use the default Streams resource manager or set up a supported external resource manager to use with Streams.

8.1.1. Setting up an external ZooKeeper server for managing Streams

An external ZooKeeper server is required for an enterprise domain. External ZooKeeper must be installed and configured before you create the domain.

Before you begin

Streams requires ZooKeeper Version 3.4.12.

Procedure

1. Install and configure ZooKeeper by using the instructions in the *ZooKeeper Administrator's Guide* on the Apache ZooKeeper website.

Note

If ZooKeeper is already installed and configured, see "Upgrading an external ZooKeeper server".

2. Configure ZooKeeper for Streams by using the following guidelines:

   - ZooKeeper runs as an ensemble of ZooKeeper servers. Ensure that there is a quorum formed for the ZooKeeper ensemble. For example, for a ZooKeeper ensemble that includes five resources, at least three resources need to be up and running to form a quorum. For reliability and availability, run ZooKeeper on at least three resources. Running ZooKeeper on five resources is preferred. For more information about setting up a ZooKeeper ensemble, see the *ZooKeeper Administrator's Guide* on the Apache ZooKeeper website.

   - For optimal performance and response time, run the ZooKeeper server on a dedicated machine, and use a dedicated local device for the transaction log. For more information, see the *ZooKeeper Administrator's Guide* on the Apache ZooKeeper website.

   - Having a supervisory process that manages each of the ZooKeeper server processes ensures that if the ZooKeeper process exits abnormally, it is restarted automatically and rejoins the cluster. For more information, see the *ZooKeeper Administrator's Guide* on the Apache ZooKeeper website.

   - Using the /tmp directory for ZooKeeper data is not recommended.

     For example, if the ZooKeeper dataDir directory is under the `/tmp` directory, an automatic system maintenance utility, such as **`tmpwatch`**, might remove files that are needed by ZooKeeper. The removal of these files might result in the failure of Streams instances that are managed by ZooKeeper.

   - If there are fsync warnings in the ZooKeeper log, the *ZooKeeper Administrator's Guide* recommends having a dedicated disk for the `dataLogDir` directory that is separate from the `dataDir` directory. Set the **`dataLogDir`** parameter in the *`ZooKeeper-installation-directory`*`/conf/zoo.cfg` file.

   - Periodically backing up the ZooKeeper `dataDir` and `dataLogDir` directories is a good practice. Recovering from backups might be necessary in case a catastrophic failure, such as a corrupted disk, occurs.

   - If you use the default ZooKeeper configuration, ZooKeeper does not remove old snapshots and log files that are stored in the data directory. To configure automatic purging of these files, you can use the **`autopurge.snapRetainCount`** and **`autopurge.purgeInterval`** parameters. For more information, see the Configuration Parameters and Maintenance sections in the *ZooKeeper Administrator's Guide* on the Apache ZooKeeper website.

   - Ensure that the value of the **`maxClientCnxns`** configuration parameter is high enough to avoid the loss of connections.

     - In the *`ZooKeeper-installation-directory`*`/conf/zoo.cfg` file, set **`maxClientCnxns=0`**. This setting removes the limit on connections.

     - If you want to set the *maxClientCnxns* parameter to some value other than *0*, periodically run the ZooKeeper *srvr* command while Streams is running. This command provides information about the number of connections.

- If the **maxClientCnxns** parameter is set to some value other than **0** and the ZooKeeper log contains warnings that there are too many connections, increase the value of the parameter for your system.

In the following **srvr** command output example, there are 1616 connections to the resource. If this number represents a typical number of connections for this system, the administrator might try a **maxClientCnxns** value that is approximately double the number of connections or **maxClientCnxns=3200**.

```
Zookeeper version: 3.4.8-1569965, built on 07/29/2014 09:09 GMT
Latency min/avg/max: 0/4/2834
Received: 14529333776
Sent: 14758752055
Connections: 1616
Outstanding: 4
Zxid: 0x952925303
Mode: follower
Node count: 237674
```

- ZooKeeper keeps data in memory and in a persistent store. The amount of data that Streams stores in ZooKeeper depends on the application runtime size. A typical amount is three times the application description language (ADL) file size. The ADL file is a configuration file that is created when a stream processing application is compiled.

  The default Java heap size for ZooKeeper is the JVM default for the system. If the maximum heap size is not sufficient for the ZooKeeper runtime system and data in memory, you can increase the size by using the JVMFLAGS environment variable. The following example shows how to set a maximum heap size of 1 GB:

  a. In the *ZooKeeper-installation-directory*/conf directory, create a java.env file.

  b. Add `export JVMFLAGS="-Xmx1024m"` to the file.

  c. Start the external ZooKeeper server.

- To avoid swapping, ensure that the Java heap size is less than the unused physical memory.

3. After you configure and start the ZooKeeper ensemble, verify that the servers in the ensemble are operating correctly by using the ZooKeeper **srvr** or **stat** command.

   In the following example, the **srvr** command is used to check a ZooKeeper ensemble that includes servers zkserver1, zkserver2, and zkserver3. The client port number is 2181.

```
$ echo srvr | nc zkserver1 2181
Zookeeper version: 3.4.8-1569965, built on 07/29/2014 09:09 GMT
Latency min/avg/max: 0/0/902
Received: 344730124
Sent: 361405124
Connections: 4
Outstanding: 0
Zxid: 0x102d77e56
Mode: follower
Node count: 10213

$ echo srvr | nc zkserver2 2181
```

```
Zookeeper version: 3.4.8-1569965, built on 07/29/2014 09:09 GMT
Latency min/avg/max: 0/0/792
Received: 331976094
Sent: 348015235
Connections: 3
Outstanding: 0
Zxid: 0x102d77e56
Mode: leader
Node count: 10213

$ echo srvr | nc zkserver3 2181
Zookeeper version: 3.4.8-1569965, built on 07/29/2014 09:09 GMT
Latency min/avg/max: 0/0/792
Received: 353974562
Sent: 370803297
Connections: 3
Outstanding: 0
Zxid: 0x102d77e56
Mode: follower
Node count: 10213
```

Related tasks:

"Creating a Streams enterprise domain"

You can create an enterprise domain by using the Streams graphical user interfaces or the *streamtool* command-line interface.

Related reference: Troubleshooting ZooKeeper problems

Use these solutions to resolve problems that you might encounter with external ZooKeeper or embedded ZooKeeper.

8.1.1.1. Upgrading an external ZooKeeper server

Use this procedure to upgrade the external ZooKeeper servers that are used by Streams enterprise domains.

About this task

The upgrade procedure that you use depends on how your ZooKeeper ensemble is configured.

- *Rolling upgrade*: If each ZooKeeper server is running on a dedicated machine and the hostname or zkconnect string is not changing, you can upgrade the server while the Streams domains are running.
- *Standard upgrade*: If the ZooKeeper servers share a machine or there are changes to the hostname or zkconnect string, you must perform a standard upgrade on each server. Streams domains must be stopped before performing the upgrade.

Procedure

For a *rolling upgrade*, complete the following steps:

1. Stop any ZooKeeper watchdog or monitor that restarts the ZooKeeper server.
2. Verify that the ZooKeeper ensemble is up and running by using the ZooKeeper **`srvr`** or **`stat`** command. For a **`srvr`** command example, see the ZooKeeper setup procedure.

3. Stop one of the ZooKeeper servers that is defined in the zoo.cfg file and verify that a quorum still exists.

   In the following example, the ZooKeeper ensemble includes zkserver1, zkserver2 and zkserver3. The client port number is 2181.

   a. To stop zkserver1, enter the following command on zkserver1:

      ```
      ZooKeeper-installation-directory/bin/zkServer.sh stop
      ```

   b. To verify that a quorum still exists, use the *srvr* command and specify another server in the ZooKeeper ensemble on the command, for example:

      ```
      echo srvr | nc zkserver2 2181
      ```

      If a quorum exists, the **srvr** command output includes information about the ZooKeeper version, latency, and other information as shown in the following example:

      ```
      Zookeeper version: 3.4.8-1569965, built on 08/11/2016 09:09 GMT
      Latency min/avg/max: 0/4/2834
      Received: 14529333776
      Sent: 14758752055
      Connections: 1616
      Outstanding: 4
      Zxid: 0x952925303
      Mode: follower
      Node count: 237674
      ```

4. Back up the following directories on the server:

   - `dataDir`
   - `dataLogDir`, if specified in the `zoo.cfg` file

5. Perform the upgrade on the server.

6. Copy the content in the following directories to the upgraded server:

   - `dataDir`
   - `dataLogDir`, if specified in the `zoo.cfg` file

7. If applicable, transfer the following files to the upgraded server:

   - `zoo.cfg`
   - `java.env`
   - `log4j.properties`

8. Verify that the zoo.cfg file references the correct file path for the `dataDir` and `dataLogDir` directories, and that the `myid` files are configured correctly.

9. Restart the upgraded ZooKeeper server, for example:

   ```
   ZooKeeper-installation-directory/bin/zkServer.sh start
   ```

10. Repeat Steps 2-9 for the other ZooKeeper servers in the ensemble.

11. After all servers in the ensemble are upgraded, restart the ZooKeeper watchdog or monitor, if applicable.

For a *standard upgrade*, complete the following steps:

1. Stop the Streams domains that are using the ZooKeeper ensemble. To stop a domain, enter the following command:

   ```
   streamtool stopdomain -d domain-id
   ```

2. Log in as user root on each Streams resource that is running the domain controller service as a Linux system service, and enter the following command:

   ```
   streamtool stopdomainhost -d domain-id
   ```

3. Stop any ZooKeeper watchdog or monitor that restarts the ZooKeeper server.

4. Stop the ZooKeeper servers. To stop a ZooKeeper server, enter the following command on the server:

   ```
   ZooKeeper-installation-directory/bin/zkServer.sh stop
   ```

5. Back up the following directories on each server:

   - `dataDir`
   - `dataLogDir`, if specified in the `zoo.cfg` file

6. Perform the upgrade on each server.

7. If applicable, transfer the following files to each server:

   - `zoo.cfg`
   - `java.env`
   - `log4j.properties`

8. Verify that the zoo.cfg file references the correct file path for the `dataDir` and `dataLogDir` directories, and that the `myid` files are configured correctly for each server.

9. Copy the content in the following directories to each server:

   - `dataDir`
   - `dataLogDir`, if specified in the `zoo.cfg` file

10. Restart each of the upgraded ZooKeeper servers. To start a ZooKeeper server, enter the following command on the server:

    ```
    ZooKeeper-installation-directory/bin/zkServer.sh start
    ```

    When a quorum is formed, the data are automatically replicated to the servers in the quorum.

11. Verify that the ZooKeeper ensemble is up and running by using the ZooKeeper **`srvr`** or **`stat`** command. For a **`srvr`** command example, see the ZooKeeper setup procedure.

12. After all servers in the ensemble are upgraded, restart the ZooKeeper watchdog or monitor, if applicable.

13. Log in as user root on each Streams resource that is running the domain controller service as a Linux system service, and complete the following steps for each domain:
    a. Update the **zkconnect** property in the following file to the proper value:

       `/etc/InfoSphere_Streams/domain-id/controller.properties`

    b. Start the domain controller service on the resource by entering the following command:

       `streamtool startdomainhost -d domain-id`

14. Start the Streams domains. To start a domain, enter the following command:

    `streamtool startdomain -d domain-id`

Related reference: Streamtool commands

This reference section provides a description of each *streamtool* command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man command-name`.

8.1.2. Setting up the default user authentication method for a Streams enterprise domain

Before you create a Streams enterprise domain, use these guidelines to set up an LDAP server or the PAM authentication service for user authentication. After you create the domain, you can customize user authentication.

When you create an enterprise domain, you must specify either LDAP or PAM as the default user authentication method for Streams. After creating the domain, you can use the following additional options to customize user authentication:

- Login module authentication
- Client certificate authentication
- Kerberos authentication :

Related tasks

"Customizing user authentication for a Streams enterprise domain"

When you create an enterprise domain, you must specify either LDAP or PAM as the default user authentication method for Streams. After creating the domain, you can use login modules, client certificates, and Kerberos to customize user authentication.

8.1.2.1. Setting up an LDAP server for authenticating Streams users

The preferred authentication service for user authentication in a Streams enterprise domain is the Lightweight Directory Access Protocol (LDAP) authentication service. If you are using LDAP for user authentication, it must be installed and configured before you create an enterprise domain.

Procedure

1. The system administrator installs and configures LDAP by using the instructions in the LDAP documentation.

2. Obtain the LDAP property values for Streams.

Important

- You must specify the **`serverUrl`** and **`userDNPattern`** property values when you create an enterprise domain.
- The group, user, and secondary lookup properties are optional.
  - The group and user properties control how user groups are searched on the LDAP server and how Streams determines which groups have a user as a member.
  - Streams uses the secondary lookup property to perform an LDAP query to find the LDAP username for the provided username, and then authenticates at the LDAP server with this LDAP username and the provided password.
- The group, user, and secondary lookup functions that are used by Streams no longer require that the LDAP server allow anonymous binds.
- If you use an LDAP server that does not enable anonymous binds, Streams uses the credentials that are specified on the **`security.ldapAdministratorUser`** and **`security.ldapAdministratorPassword`** domain properties when it runs LDAP queries during the authentication process. You can specify these property values when you create the domain or after the domain is created by using the**`streamtool setldapadminconfig`** command. For more information about these properties, enter `streamtool man domainproperties`. For more information about the command, enter `streamtool man setldapadminconfig`.

The following **`streamtool mkdomain`** command example shows the LDAP options that are specified on the command:

```
streamtool mkdomain -d sampledomain --zkconnect myzookeeper:2181
--ldap --server-url "ldap://ldap1.ibm.com:389"
--user-dn "cn=*,ou=people,dc=ibm,dc=com"
--group-obj groupOfNames --group-attr member --user-attr uid
--group-srch "ou=group,dc=ibm,dc=com"
--user-secondary-lookup "(&(objectclass=person)(uidNumber=*)) uid"
--property security.ldapAdministratorUser=admin1
--property security.ldapAdministratorPassword=password
```

serverUrl

LDAP server URL. This URL includes the host name and port number of the LDAP server, for example, `ldap://ldap1.ibm.com:389`.

userDnPattern

User DN Pattern. This pattern is used to create a distinguished name (DN) for a user during login, for example:

`cn=*,ou=people,dc=ibm,dc=com`, which is valid for any LDAP server type.

`ADDOMAINNAME\\*`, which is valid for Windows Active Directory only.

When the user logs in, the user ID is substituted for the asterisk (*) in the pattern.

groupObjectClass

LDAP group object class that is used to search for group names.

groupSearchBaseDn

LDAP base DN that is used to search for groups.

groupAttributeWithUserNames

LDAP name of the element in the group record that contains the list of members in the group.

userAttributeStoredInGroupAttribute

LDAP name of the element in a user record that is stored in the group record.

userSecondaryLookup

LDAP user secondary lookup query that Streams uses to find the LDAP user name from the specified user ID, for example:

```
(&(objectclass=ibmperson)(notesshortname=*)) uid"
```

.

Example

The following examples show how to use LDAP search commands to validate the following options for the **`streamtool mkdomain`** command:

- **`--server-url`**: Specifies the URL for the LDAP server that the domain uses to authenticate users.
- **`--user-dn`**: Specifies the LDAP pattern that is used to create a DN when the user logs in. For example: "cn=*,ou=People,ou=streams,o=ibm.com®". When the user logs in, their user ID is substituted for the asterisk (*) in the pattern.
- **`--user-secondary-lookup`**: Specifies an LDAP user secondary lookup query, which Streams uses to find the LDAP user name from the provided user name.

Note

Your values will be different in the following examples. Your LDAP server administrator who controls and manages the LDAP schema can provide the appropriate strings to use in the LDAP searches.

The LDAP server in this example authenticates John Doe by using his uid, which is the serial number plus the country code.

John Doe
serial number: 123456
country code: 897
uid: 123456897
Notes® short name: jdoe

In the following LDAP search, John Doe's record is looked up by using `objectclass=ibmperson` and `uid=123456897`, which is a combination of the serial number and country code.

```
ldapsearch -xLLL -h bluepages.ibm.com:389 -b "c=us,ou=bluepages,o=ibm.com"
"(&(objectclass=ibmperson)(uid=123456897))"
```

Result:

```
dn: uid=123456897,c=us,ou=bluepages,o=ibm.com
objectclass: person
objectclass: organizationalPerson
objectclass: ibmPerson
objectclass: ePerson
objectclass: top
ou: bluepages
o: ibm.com
ibmserialnumber: 123456
employeecountrycode: 897
notesemail: CN=John Doe/OU=Rochester/O=IBM@IBMUS
notesmaildomain: IBMUS
notesmailfile: mail1\jdoe
notesshortname: jdoe
co: USA
uid: 123456897
```

In the following LDAP search, the data is filtered by `objectclass=ibmperson` and `notesshortname=jdoe`, and the `uid` is returned.

```
ldapsearch -xLLL -h bluepages.ibm.com:389 -b "c=us,ou=bluepages,o=ibm.com"
(&(objectclass=ibmperson)(notesshortname=jdoe))" uid
```

Result:

```
dn: uid=123456897,c=us,ou=bluepages,o=ibm.com
uid: 123456897
```

Domain users know their notesshortname but might not know their uid, which is a combination of their serial number and country code. For users to log in with their notesshortname, the following **`streamtool mkdomain`** command can be used to create the enterprise domain. The previous ldapsearch example verified the value to use for the **`--user-secondary-lookup`** parameter.

```
streamtool mkdomain -d jhbMiniCluster --ldap --server-url "ldap://
bluepages.ibm.com:389"
--user-dn "uid=*,c=us,ou=bluepages,o=ibm.com"
--user-secondary-lookup "(&(objectclass=ibmperson)(notesshortname=*)) uid"
 --owner jdoe
```

Result

```
User jdoe LDAP password:*********
CDISA0017I Creating the following domain: jhbMiniCluster.
CDISA0018I The following domain was created successfully: jhbMiniCluster.
```

Related tasks:

“Creating a Streams enterprise domain”

You can create an enterprise domain by using the Streams graphical user interfaces or the **`streamtool`** command-line interface.

“Customizing user authentication for a Streams enterprise domain”

When you create an enterprise domain, you must specify either LDAP or PAM as the default user authentication method for Streams. After creating the domain, you can use login modules, client certificates, and Kerberos to customize user authentication.

Related reference:

Streamtool commands

This reference section provides a description of each *streamtool* command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man command-name`.

8.1.2.2. Setting up the PAM authentication service for Streams users

The Pluggable Authentication Module (PAM) authentication service is the default mechanism for user authentication on Linux systems. You can create a Streams enterprise domain that uses the default Linux authentication option, which is PAM with a UNIX backend, or you can use other PAM options such as PAM with the LDAP backend. The preferred PAM option for a high availability environment is PAM with the LDAP backend.

Before you begin

If you configure security to use PAM with a UNIX backend, only the domain owner can log in to a Streams domain by using a user ID and password. This restriction does not apply to domains that use other PAM options such as PAM with the LDAP backend.

You can use the following options to work around this restriction:

- Use the **`security.runAsRoot`** domain property to enable both the domain owner and non-domain-owner clients to log in with a user ID and password. You can update this property by using the **`streamtool setdomainproperty`** command. The following example shows how to update the property by using the command:

  ```
  streamtool -d domain1 setdomainproperty security.runAsRoot=true
  ```

  Note

  This property pertains only to resources where the domain controller service is registered as a Linux system service. You can register the domain controller service as a system service by running the **`streamtool registerdomainhost`** command.

- The **`security.runAsRoot`** property works for all Streams clients, including the Streams Console, Streams Studio, the **`streamtool`** command-line interface, and the REST and JMX domain management API clients.

  Note

  The **`security.runAsRoot`** property must be used to enable Streams Studio, and the REST and JMX domain management API clients to use the domain and its instances.

- The Streams Console can be configured to use certificate based client authentication as an alternative to the **`security.runAsRoot`** property.

- Public and private keys can be used by the **streamtool** command-line interface as an alternative to the **security.runAsRoot** property.

Procedure

The system administrator configures PAM by using the instructions in the PAM documentation.

Important

The Streams resource that is running the authentication and authorization service must be able to access the PAM backend to verify and authenticate Streams users.

- For PAM with a UNIX backend, the Streams users must be defined on this system.
- For PAM with the LDAP backend, the LDAP server must be accessible from the resource that is running the authentication and authorization service.

Related concepts:

"User authentication options for Streams"

The default user authentication method for Streams domains is PAM or LDAP. For a basic domain, Streams uses PAM. For an enterprise domain, you can specify either LDAP or PAM as the default method when you create the domain, and then customize user authentication after the domain is created.

Related reference:

streamtool setdomainproperty command streamtool registerdomainhost command

8.1.3. Setting up an external resource manager for a Streams enterprise domain

Streams supports user-defined resource managers. Using an external resource manager with Streams is optional.

8.1.3.1. Setting up a user-defined external resource manager for Streams

If you plan to use a user-defined external resource manager to manage resources, use this procedure to configure your resource manager for Streams.

Procedure

1. Create a Streams enterprise domain and specify the user-defined resource manager as the external resource manager for the domain.

   When you create the domain by using the **streamtool mkdomain** command, specify `--property domain.externalResourceManager=`*`user-defined-value`* on the command.

2. Start the resource manager.
3. Start the Streams domain by using the **streamtool startdomain** command.

Related concepts:

Resources

Related tasks:

"Creating a Streams enterprise domain"

You can create an enterprise domain by using the Streams graphical user interfaces or the *streamtool* command-line interface.

"Setting up resources in a Streams enterprise domain"

Administering domains

Related reference:

Streamtool commands

This reference section provides a description of each *streamtool* command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man command-name`.

8.2. Creating a Streams enterprise domain

You can create an enterprise domain by using the *streamtool* command-line interface.

Before you begin

- Set up an external ZooKeeper server to manage Streams.
- Set up the default user authentication method for the domain.
- If you are using an external resource manager, set up the external resource manager for Streams.

Creating an enterprise domain by using the Streams streamtool command-line interface

Use this procedure to create an enterprise domain by using the Streams **`streamtool mkdomain`** command.

Procedure

1. To set the Streams ZooKeeper connection string in an environment variable, enter the following command. This setting eliminates the need to specify the connection string on subsequent commands.

   ```
   export STREAMS_ZKCONNECT=external-ZooKeeper-connection-string
   ```

   For example:

   ```
   export
    STREAMS_ZKCONNECT=192.168.1.100:2181,192.168.1.101:2182,192.168.1.103:2183
   ```

2. To configure your local shell environment for Streams, enter the following command:

   ```
   source product-installation-root-directory/7.1.0.0/bin/streamsprofile.sh
   ```

3. To complete this procedure in the interactive *streamtool* interface, enter the following command:

`streamtool`

Using the interactive **`streamtool`** interface saves you time. Streams caches some command options and information so that you do not have to reenter them. Also, you do not have to specify streamtool before each command. To exit the interactive **`streamtool`** interface, enter exit or quit.

4. Create an enterprise domain by using the **`streamtool mkdomain`** command.

   - To create an enterprise domain that uses PAM for user authentication, enter the following command:

     `mkdomain -d` *`domain-id`* `--property domain.highAvailabilityCount=`*`number`*

     Notes:

     - To configure high availability in Streams enterprise domains, you must specify a **`domain.highAvailabilityCount`** domain property value greater than 1. For more information, see "Configuring high availability for Streams enterprise domains, instances, and application repositories".
     - If you are not using SSH for your communication mechanism, specify `--property domain.sshAllowed=false` on the command. By default, the **`domain.sshAllowed`** property is set to true, and Streams attempts to use SSH for communication between resources in the domain. If you are not using SSH for your communication mechanism, setting this property to false prevents unnecessary communication attempts between resources and SSH error messages.

   - To create an enterprise domain that uses LDAP for user authentication, enter the following command. When prompted, enter the LDAP password for the domain owner.

     ```
     mkdomain -d domain-id --ldap --server-url URL --user-dn DN-pattern
     --property domain.highAvailabilityCount=number
     ```

     For example:

     ```
     mkdomain -d StreamsDomain --ldap --server-url "ldap://
     ldap.streams.com:389"
     --user-dn "uid=*,ou=people,dc=streams,dc=com" --property
      domain.highAvailabilityCount=3
     ```

     Notes:

     - Enter the **`mkdomain`** command on one line. In the examples, the lines are split for presentation purposes.
     - The owner of the domain is the user who runs the command by default.

       To specify another owner, use the following command options:

       `--owner` *`LDAP-user-ID`*

       If you specify another owner, enter the LDAP password of that owner when prompted. Otherwise, enter the LDAP password of the user who is running the command. The password

prompt tells you which user is the owner. For example, if you specify **`--owner tester1`** on the command but tester3 is creating the domain, the password prompt says User tester1 LDAP password.

- To configure high availability in Streams enterprise domains, you must specify a **`domain.highAvailabilityCount`** domain property value greater than 1. For more information, see "Configuring high availability for Streams enterprise domains, instances, and application repositories".
- If you are not using SSH for your communication mechanism, specify `--property domain.sshAllowed=false` on the command. By default, the **`domain.sshAllowed`** property is set to true, and Streams attempts to use SSH for communication between resources in the domain. If you are not using SSH for your communication mechanism, setting this property to false prevents unnecessary communication attempts between resources and SSH error messages.
- If you are using an external resource manager, set the **`domain.externalResourceManager`** domain property.
- For a user-defined external resource manager, specify `--property domain.externalResourceManager=`*`user-defined-value`* on the command.

5. To display the resources in the domain, enter the following command:

   ```
   lsavailablehosts
   ```

   As shown in the following example, there is one resource in the domain. You assign tags to this resource later in this procedure.

   ```
   Resource           Tags
   myhost1.ibm.com
   ```

6. Optional: Set up automatic recovery from failures on this resource.

   For high availability with automatic recovery from failures, the domain controller service must run as a registered Linux system service on this resource. You can set up the controller on additional resources when you add those resources to the domain.

   To set up the controller as a system service on this resource, enter the following command:

   ```
   streamtool registerdomainhost -d domain-id --zkconnect host:port
   ```

   - The `--zkconnect` option specifies the name of one or more host and port pairs for the configured external ZooKeeper ensemble. If you specify multiple host and port pairs, separate each pair with a comma. This value is the external ZooKeeper connection string. To obtain this value, you can use the **`streamtool getzk`** command.
   - You must have root authority to run this command.

7. Optional: Change the default location for Streams class cache, log, and trace files.

   The default root directory for class cache, log, and trace files that are generated by Streams is the /tmp directory. If you use a system maintenance utility such as **`tmpwatch`**, either modify the utility to exclude these files or configure a different root directory by using the following Streams domain properties:

- **`domain.fileStoragePath`**: This property specifies the root directory where Streams class cache and other temporary files are stored. If the class cache files are removed, the web management service might fail and not recover correctly.

- **`domainLog.path`**: This property specifies the root directory where Streams log and trace files are stored. If these files are removed, it might be difficult to resolve Streams problems.

You can update these properties by using the **`streamtool setdomainproperty`** command.

8. Set up additional resources in the domain.

9. If you are using the default Streams resource manager, complete the following steps after you set up the resources in the domain:

   a. Verify that the resources are added to the domain. For example, you can use the **`streamtool lsavailablehosts`** command. The following command output example shows a domain that includes three Streams resources.

   ```
   Resource       Tags
   myhost1.ibm.com
   myhost2.ibm.com
   myhost3.ibm.com
   ```

   b. Assign management and application tags to the resources. Assign at least two resources as management resources and the other resources as application resources. Tag names are case-sensitive. For example, the following commands assign the management tag to host1 and host2, and the application tag to host3:

   ```
   chhost --tags management host1,host2
   chhost --tags application host3
   ```

10. Start the domain by entering the following command:

    ```
    startdomain
    ```

    When prompted, enter your domain ID and password. If another domain owner was specified on the *mkdomain* command, enter the ID and password of the owner.

    Tip: If the domain fails to start because a port is in use, you can change the port number by using the **`streamtool setdomainproperty`** command. For example, if the domain fails to start because the management API service and web management service ports are in use, you can enter the following command. If you specify a value of 0 for the port number, Streams selects an available port.

    ```
    setdomainproperty jmx.port=0 sws.port=0
    ```

    Dynamic port allocation is not ideal, however, if you have applications that use the REST API or JMX API or use tools (such as Streams Studio or Streams Console) that access those services.

11. Create one or more instances for running stream processing applications.

12. To manage and monitor the domain, you can use the Streams Console. The domain must be started to use the console.

    To open the console:

a. Enter geturl to display the URL for the console. For example, if your resource is myhost.mydomain.com and the port number for the console is 8440, this command displays the following URL:

```
https://myhost.mydomain.com:8440/streams/domain/console
```

b. Paste this URL into your browser.

13. To exit the interactive **streamtool** interface, enter exit or quit.

Related reference:

Streamtool commands

This reference section provides a description of each **streamtool** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

8.3. Setting up resources in a Streams enterprise domain

8.3.1. Creating a domain host installation package for a Streams enterprise domain

If you are using the default Streams resource manager to manage resources in your enterprise domain, you can use this procedure to create a domain host installation package that you can use to add Streams resources to the domain.

Both Streams resources and externally managed resources can be used together in a domain.

Before you begin

- Install the main installation package for Streams on at least one resource. The main installation package includes all of the product files. To reduce the product size requirement on resources, the domain host installation package contains a subset of the Streams product files. For example, this package does not contain the toolkits.

  Note

  If you installed the main installation package on a shared file system that is accessible by all resources in the domain, you do not need to create a domain host installation package. To add resources to the domain, see "Adding Streams resources that already have the product installed".

- Optionally, create an enterprise domain. If you create the domain after you set up the resources, you must use the domain name and ZooKeeper connection string that are specified in the domain host installation package.

About this task

The domain host installation package contains a *streamsdomainhostsetup.sh* script, a response file, and a readme file. The response file contains the installation information that you specify when you create the package and helps to simplify the installation process.

The script performs the following operations:

- Installs Streams on the resource.
- Registers the resource in the domain.

  Sets up the domain controller service as a Linux system service or an unregistered service. For more information, see Options for setting up the domain controller service.
- Starts the domain controller service on the resource.

Procedure

1. Create the domain host installation package by using the Streams Console or the **`streamtool mkhostpkg`** command.

   - Streams Console

     If the domain exists, you can download the package from the Streams Console. You can specify the installation directory, installation owner, and installation group in the Streams Console Add Resources window before you download the package.

     Notes:

     - Download the package file from a resource that has the main installation package installed.

       Important: Ensure that the web management service is running on the resource, or an error message is displayed. You can use the **`streamtool getdomainstate`** command to verify that the service is running.

     - By default, the domain controller service option is selected. If selected, the controller is set up as a registered Linux system service when you install the domain host installation package. To run the controller as an unregistered service, clear the controller option check box in the Streams Console Add Resources window before you download the package.

   - **`streamtool mkhostpkg`** command

     You can create the domain host installation package by running this command even if the domain does not exist.

     By default, the **`streamtool mkhostpkg`** command uses the following options:

     - The domain ID is StreamsDomain. To use a different domain ID, specify the `-d` *`domain-id`* option on the command.
     - The generated package is located in the current directory. To use a different directory, specify the `--pkg-dir` *`pathname`* option on the command.
     - The domain controller service is set up as a registered Linux system service when you install the domain host installation package. To run the controller as an unregistered service, specify the --unregistered option on the command.

> **Note**
>
> If a non-root user runs the command with the `--unregistered` option, the domain must exist and be started.

2. Optional: Customize the response file.

   To change the options that you specified when you created the domain host installation package, extract the contents of the installation package and edit the response file. For example, you can use the same installation package to add the resource to another domain with a different ZooKeeper connection string value. Before you run the setup script on your resources, modify those properties in the response file.

Results

The domain host installation package is created. The package file is an uncompressed tar file. The contents of the files in the tar file are compressed.

What to do next

Copy the domain host installation package to all of the resources that you want to add to the domain.

After you copy the package to a resource, install and configure Streams on the resource.

Related reference:

Streamtool commands

This reference section provides a description of each **`streamtool`** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

8.3.2. Creating and deploying a resource installation package for a Streams enterprise domain

If you are using a user-defined resource manager to manage resources in your enterprise domain, you can use this procedure to create a resource installation package and deploy that package on all of the externally managed resources that you want to add to the domain.

Both Streams resources and externally managed resources can be used together in a domain.

Before you begin

- Install the main installation package for Streams on at least one resource. The main installation package includes all of the product files. To reduce the product size requirement on resources, the resource installation package contains a subset of the Streams product files. For example, this package does not contain the toolkits.

> **Note**
>
> If you installed the main installation package on a shared file system that is accessible by all resources in the domain, you do not need to create and deploy a resource installation package. To

add resources to the domain, see "Adding externally managed resources to a Streams enterprise domain".

- Set up the external resource manager for Streams.
- Requirements for installing the resource installation package:
  - The installation script does not check that the hardware, software, and other preinstallation requirements for installing Streams are satisfied. Ensure that you have satisfied these requirements before installing Streams on the externally managed resources. Otherwise, the installation might not be successful or the product might not run. For example, the installation will fail if there is not enough storage. If there are missing software dependencies, the product might not run.
  - If you run the installation script with root authority, the user ID and group that own the installation must exist before running the script. Otherwise, the script will fail and indicate that the user ID and group are missing. The root user cannot own the installed files.
  - The installation script creates the specified root installation directory and any parent directories that do not exist. You must have write permission for the last existing parent directory. For example, if you install the product in the `/opt/ibm/InfoSphere_Streams` directory and the `ibm` directory does not exist, you must have write permission for the `opt` directory. If the product version is already installed in the specified installation directory, the default behavior is that the script fails with exit code 1. For any other error, the script exits with exit code 2.
- If you are installing the package in the same root directory as a previous product version, you must be the installation owner of the previous version.

  If you are installing the package with root authority, you must specify the same installation owner as the previous version.

Procedure

1. Create the resource installation package by entering the following command:

   ```
   streamtool mkresourcepkg
   ```

   The default location of the generated package is the current directory. For a different directory, specify the *`--pkg-dir pathname`* command option.

   For more information about this command, enter `streamtool man mkresourcepkg`.

   The resource installation package contains the following files:

   - Installation script files:
     - `streamsresourceinstall.sh`: This script file is used to install the Streams product on the externally managed resources with the same options that you specified on the **`streamtool mkresourcepkg`** command.
     - `streamsresourcesetup.sh`: This script file is used to install the Streams product on the externally managed resources if you want to change any options that you specified on the **`streamtool mkresourcepkg`** command.

- `streamsresourceuninstall.sh`: This script file is used to uninstall the Streams product on the externally managed resources.

2. Deploy the resource installation package on the externally managed resources. The method for deploying the package depends on the external resource manager that you use.
   - **User-defined resource manager**: Deploy and install the package manually.
     a. Copy the package to the externally managed resources.
     b. Extract the contents of the package by entering `tar -xvf resource-installation-package-name .tar`.
3. Install the Streams product.
   - To install the product with the same options that you specified on the **`streamtool mkresourcepkg`** command, run the streamsresourceinstall.sh script file.
   - If you want to change any options that you specified on the **`streamtool mkresourcepkg`** command, run the streamsresourcesetup.sh script file.

What to do next

Add the externally managed resources to the domain.

Related reference:

streamtool mkresourcepkg

The **`streamtool mkresourcepkg`** command generates an installation package that you can use to add externally managed resources to the domain.

8.3.3. Adding resources to a Streams enterprise domain

After you install Streams on at least one resource, you can add more resources to the domain. You can obtain resources from the default Streams resource manager or an external resource manager. Streams also supports user-defined external resource managers.

Streams resources are always hosts. The externally managed resources might be other types of physical and logical entities, such as containers and process groups.

Both Streams resources and externally managed resources can be used together in a domain.

Related concepts:

Resources

8.3.3.1. Adding Streams resources to an enterprise domain

Hosts are one type of resource that you can use in Streams. They are referred to as Streams resources.

To add more Streams resources to a domain, you must install and configure Streams on the hosts.

About this task

You can copy the domain host installation package to other hosts and configure them for inclusion in the domain. Alternatively, you can install the full product on a shared file system that is accessible by each host in the domain and add those hosts as Streams resources in the domain.

Adding Streams resources by installing the domain host installation package:

Use this procedure to install and configure Streams on a resource that you are adding to a Streams domain. To reduce the product size requirement on the resource, the domain host installation package contains a subset of the Streams files.

Before you begin

The following procedure installs Streams on the resource, registers the resource with the domain, sets up the Streams domain controller service, and starts this service on the resource.

Before you start the procedure, review the following information:

- Copy the domain host installation package to this resource. For more information about how to create and customize this package, see Creating a domain host installation package.
- Create the user ID and group that owns the installation if they do not exist on this system. This ID and group are the ID and group that are specified when the domain host installation package is created. The default ID and group are streamsadmin. Streams uses this ID and group to run commands when the domain controller service processes requests on this resource.
- Disable the resource firewall. If firewall usage is required, review the firewall configuration guidelines for Streams.
- Java is required to run the preinstallation scripts in the following procedure. The default Java version that is installed with your operating system satisfies this requirement.

Procedure

1. Extract the contents of the domain host installation package by entering the following command:

   ```
   tar -xvf domain-host-installation-package-name.tar
   ```

   The *tar* command creates the StreamsDomainHost directory, which contains the **`streamsdomainhostsetup.sh`** script and other installation files.

2. Run the Streams dependency checker script.

   The user who owns the installation files runs the dependency checker script by entering the following commands:

   ```
   cd domain-host-installation-package-directory/StreamsDomainHost
   ./dependency_checker.sh
   ```

3. Correct any problems identified by the dependency checker script.

   If the dependency checker script indicates that RPMs are missing, install the RPMs by using the *yum install* command that is included with Red Hat Enterprise Linux (RHEL).

   Most of the required RPMs for Streams are in your operating system installation package. RPMs that are shipped with the product are in the main installation package, which includes all of the product files.

4. Run the Streams domain host setup script by entering the following command from the *domain-host-installation-package-directory/* `StreamsDomainHost` directory.

   ```
   ./streamsdomainhostsetup.sh
   ```

Notes:

- A root user must run the script if the domain host installation package was created with the option to run the domain controller service as a registered Linux system service. A root or non-root user can run the script if the package was created with the option to run the controller as an unregistered service.
- This script reads the responsefile.properties file in the *domain-host-installation-package-directory*/`StreamsDomainHost` directory. You can optionally change properties in the response file, such as the installation directory, before entering the following command.
- If you see the following message, review any previous warning messages. If you cannot determine the problem from the messages that are displayed, review the warnings in the summary installation logs.

  ```
  CDISI0019I The domain host setup completed with warnings.
  ```

Results

The domain host setup script performs the following operations:

- Installs Streams on the resource by running the `StreamsDomainHostSetup.bin` file.
- Registers the resource in the domain.
- Sets up the domain controller service as a registered Linux system service or an unregistered service. The controller setup depends on how the domain host installation package was created.
- Starts the domain controller service on the resource.

What to do next

Streams provides several commands that you can use to manage the domain controller service on a resource. Before using these commands, you must configure the Streams environment variables by running the following command:

```
source domain-host-installation-directory/bin/streamsprofile.sh
```

where *domain-host-installation-directory* is the root product installation directory and the product version directory. For example, if you installed Streams in the default location for user root, enter the following command:

```
source /opt/ibm/InfoSphere_Streams/7.1.0.0/bin/streamsprofile.sh
```

You can use the following commands to manage the domain controller service on a resource.

Command	Description
`streamtool getdomainhoststatus`	Displays the status of the domain controller service on the resource.

Command	Description
	You must have root authority to run this command.
streamtool startdomainhost	Starts the domain controller service on the resource. v If you set up the domain controller service to run as a registered Linux system service by running the *streamtool registerdomainhost* command, you must have root authority to run this command. v To start the domain controller service as an unregistered service, a root or non-root user can specify the --unregistered option on this command. If a non-root user runs the command with this option, the domain must exist and be started.
streamtool stopdomainhost	Stops the domain controller service on the resource. You must have root authority to run this command.
To use the following commands, the domain controller service must be running as a Linux system service. You must have root authority to run these commands.	
streamtool getdomainhostconfig	Displays the configuration properties for the domain controller service.
streamtool chdomainhostconfig	Changes the specified domain controller service properties.
streamtool rmdomainhostconfig	Removes the specified domain controller service properties.

For more information about command syntax and options, enter the following command:

```
streamtool man command-name
```

Note

If you use the sudo program, the *PATH* environment variable might not be preserved on some platforms. For example, if you run the following commands, you might see a message indicating that the *streamtool* command is not found:

```
source /opt/ibm/InfoSphere_Streams/7.1.0.0/bin/streamsprofile.sh
sudo streamtool getdomainhoststatus -d mydomain
```

```
sudo: streamtool: command not found
```

To work around this problem, enter the following commands:

```
source domain-host-installation-directory/bin/streamsprofile.sh
sudo -E $STREAMS_INSTALL/bin/streamtool getdomainhoststatus -d mydomain
```

Related concepts:

Resources

Resources are physical and logical entities that Streams uses to run services.

Related tasks:

Configuring the Streams environment by running streamsprofile.sh

Related reference:

Streamtool commands

This reference section provides a description of each **streamtool** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

Adding Streams resources that already have the product installed:

If Streams is already installed on the resource or the main installation package is installed on a shared file system that is accessible by the resource, you can use this procedure to add the resource to a Streams enterprise domain.

Procedure

Register the resource with the domain, set up the Streams domain controller service as a registered Linux system service or an unregistered service, and start the controller on the resource.

The command that you use determines how the domain controller service is set up. For more information, see Options for setting up the domain controller service.

- To set up the domain controller service as a registered Linux system service, use the **streamtool registerdomainhost** command.

  This command registers the resource with the domain, sets up the controller as a registered Linux system service, and starts the controller on the resource.

  Example:

  ```
  streamtool registerdomainhost -d domain-id --zkconnect host:port
  ```

  Notes:

  - You must have root authority to run this command.
  - You must specify a domain name and the `--zkconnect` option on this command. The domain name that you specify does not need to exist before running the command. The `--zkconnect` option specifies the name of one or more host and port pairs for the configured external ZooKeeper ensemble. If you specify multiple host and port pairs, separate each pair with a comma. This value is the external ZooKeeper connection string. To obtain this value, you can use the **streamtool getzk** command.

- To set up the domain controller service as an unregistered service, use the **streamtool startdomainhost** command.

This command registers the resource with the domain, sets up the controller as an unregistered service, and starts the controller on the resource.

Example:

```
streamtool startdomainhost -d domain-id --unregistered --
zkconnect host:port
```

Notes:

- You must specify a domain name on this command.
  - If a root user runs the command, the domain does not need to exist before running the command.
  - If a non-root user runs the command, the domain must exist and be started before running the command.
- You must specify the `--zkconnect` option on this command. The `--zkconnect` option specifies the name of one or more host and port pairs for the configured external ZooKeeper ensemble. If you specify multiple host and port pairs, separate each pair with a comma. This value is the external ZooKeeper connection string. To obtain this value, you can use the **`streamtool getzk`** command.

What to do next

Streams provides several commands that you can use to manage the domain controller service on a resource. Before using these commands, you must configure the Streams environment variables by running the following command:

```
source domain-host-installation-directory/bin/streamsprofile.sh
```

where *`domain-host-installation-directory`* is the root product installation directory and the product version directory. For example, if you installed Streams in the default location for user root, enter the following command:

```
source /opt/ibm/InfoSphere_Streams/7.1.0.0/bin/streamsprofile.sh
```

You can use the following commands to manage the domain controller service on a resource.

Command	***Description***
`streamtool getdomainhoststatus`	Displays the status of the domain controller service on the resource. You must have root authority to run this command.
`streamtool startdomainhost`	Starts the domain controller service on the resource. v If you set up the domain controller service to run as a registered Linux system service by running the *streamtool registerdomainhost* command, you must have root authority to run this command. v To start the domain controller service as an unregistered service, a root or non-root user can specify the

Command	*Description*
	--unregistered option on this command. If a non-root user runs the command with this option, the domain must exist and be started.
streamtool stopdomainhost	Stops the domain controller service on the resource. You must have root authority to run this command.
To use the following commands, the domain controller service must be running as a Linux system service. You must have root authority to run these commands.	
streamtool getdomainhostconfig	Displays the configuration properties for the domain controller service.
streamtool chdomainhostconfig	Changes the specified domain controller service properties.
streamtool rmdomainhostconfig	Removes the specified domain controller service properties.

For more information about command syntax and options, enter the following command:

```
streamtool man command-name
```

Note

If you use the sudo program, the **PATH** environment variable might not be preserved on some platforms. For example, if you run the following commands, you might see a message indicating that the **streamtool** command is not found:

```
source /opt/ibm/InfoSphere_Streams/7.1.0.0/bin/streamsprofile.sh sudo
 streamtool getdomainhoststatus -d mydomain
sudo: streamtool: command not found
```

To work around this problem, enter the following commands:

```
source domain-host-installation-directory/bin/streamsprofile.sh
sudo -E $STREAMS_INSTALL/bin/streamtool getdomainhoststatus -d mydomain
```

Related concepts:

Resources

Resources are physical and logical entities that Streams uses to run services.

Related tasks:

Configuring the Streams environment by running streamsprofile.sh

Related reference:

Streamtool commands

This reference section provides a description of each *streamtool* command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man command-name`.

Verifying requirements for Streams resources:

You can use **`streamtool`** commands to verify that Streams resources satisfy configuration and network connectivity requirements.

Procedure

- To check one or more Streams resources, use the **`streamtool checkhosts`** command. This command can check the resources before they are allocated to domains or instances. The resources that you are checking must be configured to use Secure Shell (SSH) because the command uses SSH to access each resource and run the tests.
- To check all the Streams resources in a domain, use the **`streamtool checkdomainhosts`** command.
- To check all the Streams resources in an instance, use the **`streamtool checkinstancehosts`** command.
- To check only the network configuration, specify the **`--connectivity-only`** option in any of these commands.

8.3.3.2. Adding externally managed resources to a Streams enterprise domain

To add externally managed resources to an enterprise domain, configure the resource manager to allocate resources to the domain.

Before you begin

- Set up the external resource manager for Streams.
- If you are using a user-defined resource manager, deploy the resource installation package on all of the externally managed resources that you want to add to the enterprise domain.

Note

If you installed the main installation package on a shared file system that is accessible by all resources in the domain, you can skip this step.

Procedure

1. Configure the external resource manager to allocate resources to the enterprise domain. The way that resources are allocated depends on the resource manager. For more information, see your resource manager documentation.
2. **Optional:** Update the **`domain.managementResourceAllocationTags`** domain property. The tags that you specify on this property determine which domain management services run on the same externally managed resources.

Both Streams resources and externally managed resources can be used together in a domain. Streams resources that have tags which match resource requests are used first. If additional resources are needed, Streams uses the externally managed resources.

Example: The default value of the **`domain.managementResourceAllocationTags`** property is authentication,audit,jmx,sws. For a Streams resource to be selected as a management resource, it

must have all of these tags. If the **domain.managementResourceAllocationTags** property is updated with a value of authentication,audit, a Streams resource must have both the authentication and audit tags to be selected. Additional resources will be requested for the jmx and sws tags that were removed from the property. For an additional Streams resource to be selected, it must have one or both of these tags.

For more information about domain properties, enter streamtool man domainproperties. To update domain properties, you can use the Streams Console or the **streamtool setdomainproperty** command. For information about the command, enter streamtool man setdomainproperty.

Note

You must restart the domain for the changes to take effect.

What to do next

Create instances and allocate resources to the instances.

Related tasks:

"Creating a Streams instance in an enterprise domain"

You can create an instance by using the Streams Console or the **streamtool** command-line interface.

Assigning tags to resources in a domain

8.4. Creating a Streams instance in an enterprise domain

You can create an instance by using the Streams Console or the *streamtool*

command-line interface.

8.4.1. Before you begin

1. Create and configure a Streams enterprise domain.
2. Add resources to the domain.
3. Start the domain.

8.4.2. Procedure

Use this procedure to create an instance by using the Streams *streamtool* command-line interface:

1. To configure your local shell environment for Streams, enter the following command:

source *product-installation-root-directory*/7.1.0.0/bin/streamsprofile.sh

2. To complete this procedure in the interactive *streamtool* interface, enter the following command:

streamtool

Using the interactive *streamtool* interface saves you time. Streams caches some command options and information so that you do not have to reenter them. Also, you do not have to specify streamtool before each command. To exit the interactive *streamtool* interface, enter exit or quit.

3. Create one or more instances for running stream processing applications. To create an instance, enter the following command:

mkinstance -i *instance-id* --property instance.highAvailabilityCount=*value*

--numresources *value*

For example:

mkinstance -i StreamsInstance --property instance.highAvailabilityCount=3 --numresources 3

This example requests three resources from the domain. You can allocate resources to the instance when you create it by using *--hfile*, *--hosts*, or

--numresources options. If you do not specify one of those options, the following resource specification is added when the instance starts:

--numresources 1. You can add resources later by using the *streamtool addhost* command. For more information, see "Adding resources to Streams instances".

To configure high availability in Streams instances, you must specify an *instance.highAvailabilityCount* instance property value greater than 1. For more information, see "Configuring high availability in Streams instances".

4. Start the instance by entering the following command:

startinstance

5. To exit the interactive *streamtool* interface, enter exit or quit.

8.4.3. What to do next

If you have not done so already, configure the security for your domain and instances. For example, create roles and set permissions on instance objects by using access control lists.

You can also start developing and running stream processing applications in your instance.

8.5. Configuring an instance to use dynamic resource allocation

You can configure an instance to dynamically allocate resources depending on the needs of the job. When a job is submitted, resources are acquired from the domain to support these jobs and are released when they are no longer needed.

8.5.1. Before you begin

1. Create a domain. For more information, see Creating domains.

The following sample commands create and start an enterprise domain that uses PAM for user authentication.

export STREAMS_ZKCONNECT=192.168.1.100:2181,192.168.1.101:2182

streamtool mkdomain -d myDomain streamtool genkey -d myDomain streamtool startdomain -d myDomain

2. Add Streams hosts or externally managed resources to the domain or instance to support the needs of the jobs in the instance. For more information, see Adding resources to a Streams domain or Adding resources to Streams instances .

Important: When you add a resource to a domain, make sure to consider the tagging requirements of the applications that will run in the instances in the domain. For information about using tags to restrict resources, see Using tags to restrict resources.

The following sample *streamtool* commands add a Streams host with a tag to a development environment.

streamtool adddomainhost -d MyDomain MyHost streamtool chhost --tags MyTag MyHost

The following sample *streamtool* command adds a Streams host with a tag to a production environment.

streamtool registerdomainhost -d MyDomain --tags MyTag

8.5.2. About this task

Configuring dynamic resource allocation consists primarily of setting the domain or instance properties that control it.

8.5.3. Procedure

Create the instance in the domain and set the following properties.

v *applicationResourceAllocationMode*: Set to job.

v *dynamicResourceAllocationEnabled*: Set to true.

v *jobResourceSharing*: Set to the sharing behavior (sameJob, sameUser, sameInstance) that you want to enable for the instance. For more information about the *jobResourceSharing* property, see Resource allocation for application services.

The following sample *streamttool* command creates the myInstance instance and sets the required properties. Enter the following command on the same line.

streamtool mkinstance -d myDomain -i myInstance

--property instance.applicationResourceAllocationMode=job

--property instance.dynamicResourceAllocationEnabled=true

--property instance.jobResourceSharing=sameJob

8.5.4. Results

When a job submitted to the instance, resources that match the tagging requirements are acquired from the domain to support these jobs and are released when they are no longer needed.

Remember: Dynamically allocated resources are allocated exclusively. While a resource is allocated to a specific instance, it is not available for dynamic allocation to another instance.

8.6. Adding resources to Streams instances

You can assign resources to an instance statically when you create the instance or dynamically after the instance is created. The resources are allocated to the instance from the pool of resources in the domain.

8.6.1. Before you begin

Add resources to the domain.

8.6.2. About this task

By adding more than one resource to an instance, you increase the processing power and high availability of the instance. You can tag resources to run specific application or management services and to have standby services ready on multiple resources. These standby services can take over if a failure occurs on one of the resources. For more information, see "Configuring high availability for Streams enterprise domains, instances, and application repositories".

You can use static resources that are selected when you start the instance. To allocate specific Streams hosts to an instance, use the manual resource assignment option in the Streams Console or the *--numresources* option on *streamtool* commands. To automatically allocate Streams hosts or externally managed resources when you start the instance, use the automatic resource assignment option in the Streams Console.

Tip: In general, using the automatic resource assignment option provides better allocation for high availability environments. When an automatically added resource fails, it is automatically replaced by another resource. When a manually added resource fails, the resource is not replaced.

You can also use dynamic resources that are added to an instance based on the job or processing element requirements. A dynamic resource is released from the instance when the instance is stopped or when the resource is no longer needed.

Streams provides tags that you can use to restrict access to a resource. This can be helpful if some resources have special capabilities, such as access to a private data source. For more information, see Using tags to restrict resources.

8.6.3. Procedure

1. Create an instance. When you create the instance, you can optionally include resource specifications.

Note: To dynamically allocate resources, you must set the associated properties when you create the instance. For more information, see Configuring an instance to use dynamic resource allocation.

2. Optional: Specify which instance management services run on the same externally managed resources by updating the *instance.managementResourceAllocationTags* instance property.

Use the Streams Console or the *streamtool setproperty* command to update the property value. You must restart the instance for the changes to take effect. For more information, run *streamtool man properties*.

3. Optional if dynamic resource allocation is enabled: Add or change the resource specifications for the instance.

    a. Use the Streams Console to edit the instance and change its resource specifications. Alternatively, use the following *streamtool* commands: *addresourcespec*, *chresourcespec*, *rmresourcespec*, *addhost*, and *rmhost*.

    b. Optional: Configure resource restrictions. For instructions, see Using tags to restrict resources.

8.6.4. Results

Static resources are allocated to the instance when it starts. If your instance has unfilled resource requests, resources are added to the instance whenever an appropriate resource becomes available in the domain. You do not need to stop and restart the instance.

If dynamic resource allocation is enabled, resources are acquired from the domain to support submitted jobs and are released when they are no longer needed.

Related reference:

Streamtool commands

This reference section provides a description of each *streamtool* command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

8.7. Configuring high availability for Streams enterprise domains, instances, and application repositories

Both domains and instances have a high availability count property. This property indicates how many copies of management services are maintained by a domain, instance, or application repository. The extra copies (*standby services*) allow for automatic failover if resources unexpectedly fail or become unavailable.

8.7.1. Configuring high availability in Streams enterprise domains

If you configure high availability for the management services in a domain, Streams can switch to using standby services in the event of a software, hardware, or network interruption.

Before you begin

For high availability with automatic recovery from failures, the domain controller service must be running as a Linux system service on the Streams resource. For more information, see Options for setting up the domain controller service on Streams resources.

About this task

To configure high availability for a domain, set the **`domain.highAvailabilityCount`** domain property to a value greater than one. This property specifies the total number of each Streams domain

management service. For example, a value of three means that there is one active and two standby services for each management service in the domain. Streams can switch to using standby services in the event of a software, hardware, or network interruption. For more information about this domain property, run **`streamtool man domainproperties`**.

Note

The web management service is not affected by the **`domain.highAvailabilityCount`** setting because that service does not have any standby services. The web management service can run on resources that are tagged as sws or management resources.

To affect the placement of active and standby services on the resources in your domain, tag the resources. For example, management services for the domain run on resources that have the following management tag. If you prefer to run specific management services on specific resources, use the appropriate tags for those services. For example, the authentication and authorization service runs on resources that have the following authentication tag.

Note

There can be only one management API service for each resource. The standby services for the management API service can run on other resources that are tagged as jmx or management resources.

For more information about the Streams services and tags, see Domain and instance services.

Procedure

- To change the **`domain.highAvailabilityCount`** domain property value, use the **`streamtool setdomainproperty`** command. You can also specify that property value when you create the domain.
- To change the tags that are associated with resources, use the Streams Console or the **`streamtool chhost`** command. For more information, see Assigning tags to resources in a domain.

8.7.2. Configuring high availability in Streams instances

If you configure high availability for the management services in an instance, Streams can switch to using standby services in the event of a software, hardware, or network interruption.

Before you begin

For high availability with automatic recovery from failures, the domain controller service must be running as a Linux system service on the Streams resource. For more information, see Options for setting up the domain controller service on Streams resources.

About this task

To configure high availability for an instance, set the **`instance.highAvailabilityCount`** instance property to a value greater than one. This property specifies the total number of each Streams instance management service. For example, a value of three means that there is one active and two standby services for each management service in the instance.

Streams can switch to the standby services in the event of a software, hardware, or network interruption.

A standby service is not created for the application deployment service on each application resource, however, the application deployment service is restarted in the event of a software, hardware, or network interruption. There can only be one application service for each resource, even if the **`instance.highAvailabilityCount`** is set to a higher value. For more information about this instance property, run **`streamtool man properties`**.

To affect the placement of active and standby services on the resources that are allocated to your instance, tag the resources. For example, application services run on resources with the application tag, management services run on resources with management tag, and view services run on resources with the view tag. For more information about the Streams services and tags, see Domain and instance services.

Procedure

- To change the **`instance.highAvailabilityCount`** instance property value, use the Streams Console or the **`streamtool setproperty`** command. You can also specify that property value when you create the instance.
- To change the tags that are associated with resources, use the Streams Console or the **`streamtool chhost`** command. For more information, see Assigning tags to resources in a domain .

8.7.3. Configuring high availability in Streams for an application repository

If you configure high availability for an application repository, this application repository should be hosted on highly available storage. You can set the application repository at the domain level or at an individual instance level. If you configure high availability for an instance, this instance level setting takes priority over the domain level setting.

Before you begin

For high availability with automatic recovery from failures, the domain controller service must be running as a Linux system service on the Streams resource. For more information, see Options for setting up the domain controller service on Streams resources.

About this task

If the hosts or resources of an application, which is configured as highly available, crash, Streams pulls this application out of the application repository and starts the processing elements (PEs) of this application on new hosts.

If the property is not set, Streams uses by default the file system repository of the **`applicationBundlesPath`** property to determine the location of the application directory.

Procedure

- The following example configures an application repository that is running on a highly available **shared file system**. The configuration is set at the **instance level**.

  The **`streamtool setproperty`** command sets the property values for an instance. You have to update the values in this command:

```
streamtool setproperty
 instance.repositoryConfigurationApplicationCache="{\"type\":
\"filesystem\",\"baseDirectory\":\"/mydirectory/apprepos\"}"
```

- The following example configures an application repository that uses the **IBM Cloud Object Storage**. The configuration is set at the **domain level**. You have to update the values in this command:

  ```
  streamtool setrestrictedconfig cosconfig="{\"IBMCOSaccessKey\":
  \"theaccesskeygoeshere\",\"IBMCOSsecretKey\":\"thesecretykeygoeshere\"}"
  ```

  The **`streamtool setdomainproperty`** command sets the property values for the domain. You have to update the values for *`endpoint`*, *`bucket`*, and *`auth`*:

  ```
  domain.repositoryConfiguration="{\"type\":\"ibmcloudobjectstorage\",
  \"endpoint\"
  :\"theendpointforCOSgoeshere\",\"bucket\":\"apprepos\",\"auth\":
  \"cosconfig\"}"
  ```

Note

auth

Is the name of the restricted configuration created in the prior step.

bucket

If you have not created or defined a bucket using the Cloud Object Storage (COS) service, a bucket is created for you.

Chapter 9. Configuring security for Streams

Each Streams domain and instance maintains its own security configuration.

9.1. User authorization for Streams

Streams uses access control lists (ACLs) to manage user authorization for domains and instances. An ACL contains the type of domain and instance objects to secure and the actions that a user or group is authorized to perform against the object. The ACLs are initialized when you create a domain or instance.

9.1.1. Security objects and access permissions for Streams domains and instances

Streams security objects and permissions enable you to control access to domain and instance resources and data. You can update security objects and access permissions by using the Streams Console or **`streamtool`** commands.

Security objects are hierarchical in nature, in that some objects are included by other objects. For example, a jobs object can include multiple jobgroup objects, which include job_*id* objects for each job that is running in the system.

Each object is assigned an access permission and a default permission:

- The *access permission* identifies which users, groups, or roles have permission to perform operations against this type of object.
- The *default permission* identifies the set of permissions that are granted to new child objects created under this object. Default permissions are only important for the jobs security object when you create new job groups.

You can set the access and default permissions by using the Streams Console or the following **`streamtool`** commands:

- **`streamtool setdomainacl`** for domains
- **`streamtool setacl`** for instances

The permissions for a submitted job are contained in its job group and are changed by using the **`streamtool grantjobpermission`** and **`streamtool revokejobpermission`** commands. The name of the default job group for an instance is default.

Streams determines the permissions for a user from the following types of permissions:

- Specific user permissions
- Role permissions
- Group permissions

Therefore, removing permissions for a user might require that you remove role or group permissions in addition to the specific user permissions. You might also need to remove the user and their groups

from one or more roles. For more information, see the example that shows how to remove security permissions for a Streams user.

Related concepts:

"Roles"

A *role* is a collection of permissions that can be assigned to a user or group of users.

"Job groups"

A *job group* is a group of jobs that have the same authority or permissions.

Related tasks:

"Viewing and configuring ACL settings for Streams domains and instances"

You can review and configure the access control list (ACL) settings for Streams domains and instances by using the **streamtool** command-line interface. You can also grant access privileges to domain and instance resources by using the Streams Console.

Administering a domain in the Streams Console

You can use the Streams Console to monitor and manage the Streams instances and applications in your domain from any computer that has HTTP connectivity to the server that is running the web management service.

Related reference:

Streamtool commands

This reference section provides a description of each **streamtool** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

Security objects for Streams domains and instances

This information describes the security objects for Streams domains and instances.

Table 9.1. Streams security objects for domains

Domain object	Parent object	Description
domain	Not applicable	Controls who is allowed to start, stop, or control a domain.
config	domain	Controls who is allowed to change the configuration for the domain
hosts	domain	Controls who is allowed to view, add, and remove hosts from the domain configuration.
instances	domain	Controls who is allowed to view, add, and remove instances from the domain configuration.
system-log	domain	Controls who has access to view the domain and host log data.
appconfig	domain	Controls who has access to view, add, and remove domain-level application configurations.

Domain object	Parent object	Description
appconfig_<*element-name*>	appconfig	Controls who has access to view, add, and remove domain-level application configuration security elements.

Table 9.2. Streams security objects for instances

Instance object	Parent object	Description
instance	Not applicable	Controls who is allowed to start, stop, or view an instance.
config	instance	Controls who is allowed to change the configuration for the instance.
hosts	instance	Controls who is allowed to view, add, and remove hosts from the instance configuration.
jobs	instance	Controls who is allowed to submit new jobs to the instance.
jobgroup_*name*	jobs	Controls who is allowed to view or change all jobs submitted in the job group for the running instance, and who can submit a job in the instance. The parent job group is checked for authority when checking that a user has permission to submit a job. Updated permissions for a job group are used when checking permissions for a submitted job in the running instance. Job group permissions can be updated by using the **`streamtool grantjobpermission`**, **`streamtool revokejobpermission`**, or **`streamtool setacl`** command. The Access Control List entries for jobgroup_*name* objects control the export of data from a job and the import of data to a job. In order for one job to export data to a second job, the user that started the exporting job must have write access to the second job. In order for the second job to import data from the exporting job, the user that started the importing job must have read access to the exporting job. If permissions are changed after a job is submitted, either the exporting or importing PE must be restarted for the changes to take effect.
jobs-override	instance	Controls who is allowed to override the resource load protection settings when submitting jobs.
application-log	instance	Controls who is allowed to view the application log data.
system-log	instance	Controls who has access to view the instance and host log data.
appconfig	instance	Controls who has access to view, add, and remove instance-level application configurations.
appconfig_<*element-name*>	appconfig	Controls who has access to view, add, and remove instance-level application configuration security elements.

Access permissions for Streams domain and instance objects

This information describes the administrator and user access permissions for Streams domain and instance objects.

Some objects require multiple permissions. For example, you must have search and delete authority for the hosts domain object to remove a resource from a domain.

The person who created each listed object is the owner. Owners have 'own' permission, which grants them all rights on the object.

Table 9.3. Administrator and user access permissions for domain objects

Domain object	DomainAdministrator permissions	DomainUser permissions
domain	delete: Remove a domain.	No access permissions
	write: Start and stop a domain; restart domain services; and clean domain logs.	
config	read: View tags and properties.	read
	write: Manage roles, users, and groups; modify properties; view and set permissions; and make and modify tags.	
hosts	add: Add a resource to the domain.	search, read
	read: Get information about the state of the domain.	
	search: View available resources.	
	search + delete: Remove a resource from the domain.	
	search + read + write: Modify tags.	
	write: Quiesce and resume resources, and remove tags.	
instances	add: Create or copy an instance.	search
	search: View the instances in the domain.	
system-log	read: View and get the domain, instance, and job logs.	No access permissions
appconfig	search: View domain-level application configurations.	search
	add: Create new application configurations at the domain level.	
appconfig_*element-name* where *element-name* is the name of the application configuration object within the domain configuration	read: View domain-level application configuration elements.	read
	write: Create an application configuration element at the domain level.	
	delete: Remove a domain-level application configuration element.	

Table 9.4. Administrator and user access permissions for instance objects

Instance object	DomainAdministrator and InstanceAdministrator permissions	InstanceUser permissions
instance	delete: Remove an instance.	search

Instance object	DomainAdministrator and InstanceAdministrator permissions	InstanceUser permissions
	search: Capture information about the resources in an instance; copy an instance; list the instance when listing domain instances; view the instance and application logs; and get information about the state of an instance.	
	write: Start and stop the instance, and clean the instance log.	
config	read: View properties and list job groups.	read
	write: Manage roles, users, and groups; modify properties; view and set permissions; create and remove job groups; and manage resource specifications.	
hosts	add: Add a resource to an instance.	search, read
	read: Get information about the state of the instance.	
	search + delete: Remove a resource and its services from an instance.	
	search + read: List resources in an instance.	
	search + read + delete: Remove a resource specification from an instance.	
	search + read + write: Modify a resource specification.	
	write: Quiesce and resume resources.	
jobs	add: Create a job group.	search, add
	search: List jobs and PEs, and get information about the state and status of jobs and PEs.	
default job group or any job group created by a user	add: Submit a job.	add The user who submits a job has the following permissions: read, write, add, delete
	delete: Stop a job or a PE in a job.	
	add + delete: Update or restart a PE in a job.	
	read: Receive data from jobs; get information about the state of jobs; list jobs and PEs; and view application and PE logs.	
	write: Send data to jobs.	
jobs-override	add: Run the **`streamtool submitjob`** command with the **`--override`** option.	No access permissions
application-log	read: View and get the application logs.	read
system-log	read: View and get the instance and application logs.	No access permissions
appconfig	search: View instance-level application configurations.	search, add

Instance object	DomainAdministrator and InstanceAdministrator permissions	InstanceUser permissions
	add: Create application configurations at the instance level.	
appconfig_*element-name* where *element-name* is the name of the application configuration object within the instance configuration	read: View instance-level application configuration elements.	read, write, delete
	write: Create an application configuration element at the instance level.	
	delete: Remove an instance-level application configuration element.	

Related concepts:

"Job groups"

A *job group* is a group of jobs that have the same authority or permissions.

"Roles"

A *role* is a collection of permissions that can be assigned to a user or group of users.

Related reference:

"Streamtool commands that require user authorization"

The **streamtool** command description provides information about the authority that is required to run each command.

Example 9.1. Removing security permissions for Streams users

Removing the security permissions for a user might require that you remove role or group permissions in addition to the specific user permissions. This example shows you how to remove the security permissions for a Streams user by removing both specific user permissions and role permissions.

Procedure

1. Create a Streams domain and instance, and start the domain. For example:

   ```
   export STREAMS_ZKCONNECT=external-ZooKeeper-connection-string
   streamtool mkdomain -d domain1 --property sws.port=0 --property jmx.port=0
   streamtool genkey -d domain1
   streamtool startdomain -d domain1
   streamtool mkinstance -d domain1 -i instance1
   ```

   For more information about creating domains and instances, see Chapter 8, "Setting up a Streams enterprise domain on multiple resources,".

2. To complete this procedure in the interactive **streamtool** interface, enter the following command:

   ```
   streamtool
   ```

   Using the interactive **streamtool** interface saves you time. Streams caches some command options and information so that you do not have to reenter them. Also, you do not have to specify streamtool before each command. To exit the interactive **streamtool** interface, enter exit or quit.

3. Create the Example role.

```
mkrole -d domain1 -i instance1 Example
CDISC0154I The Example role was created for following instance: instance1.
The instance is in the domain1 domain.
```

4. Add the user to the Example role. You do not have to reenter the domain and instance name on this and subsequent commands because this information is cached.

```
adduserrole Example user1
CDISC0156I The Example role was assigned to the following user: user1.
The role applies to the instance1 instance in the domain1 domain.
```

5. View the access control list (ACL) for the config instance security object.

```
getacl config
# object: config
# parent: instance
# owner: DomainAdministrator
# persistent: yes
user:admin1:rw---o
role:InstanceUser:r-----
role:DomainAdministrator:rw---o
role:InstanceAdministrator:rw---o
```

6. View the permissions for `user1`. Note that `user1` has no permissions.

```
lspermission user1
application-log:------
config:------
hosts:------
instance:------
jobgroup_default:------
jobs:------
jobs-override:------
system-log:------
```

7. Configure read and write permission on the `config` instance security object for `user1`.

```
setacl user:user1:rw config
CDISC0019I The access control list for the instance1 instance in the domain1 domain was
 updated.
```

8. Configure read and write permission on the config instance security object for the Example role.

```
setacl role:Example:rw config
CDISC0019I The access control list for the instance1 instance in the domain1 domain was
 updated.
```

9. View the permissions for user1. Note that user1 has read and write (rw) permission on the config instance security object.

```
lspermission user1 application-log:------
config:rw----
hosts:------
instance:------
jobgroup_default:------
jobs:------
jobs-override:------
system-log:------
```

10. Remove read and write permission on the `config` instance security object for `user1`.

```
setacl user:user1-rw config
```

```
CDISC0019I The access control list for the instance1 instance in the domain1 domain was
 updated.
```

11. View the permissions for `user1`. Note that the read and write permission on the `config` instance security object is not removed for `user1`.

```
lspermission user1 application-log:------
config:rw----
hosts:------
instance:------
jobgroup_default:------
jobs:------
jobs-override:------
system-log:------
```

12. View the ACL on the `config` instance security object and note that the Example role has read and write permission.

```
getacl config
# object: config
# parent: instance
# owner: DomainAdministrator # persistent: yes user:admin1:rw o
role:InstanceUser:r-----
role:Example:rw----
role:DomainAdministrator:rw o
role:InstanceAdministrator:rw o
```

13. List the instance roles and note that `user1` is a member of the Example role.

```
lsrole
Role: Example
  Users: user1
  Groups:
Role: InstanceAdministrator
  Users: admin1
  Groups:
Role: InstanceUser
  Users:
  Groups:
```

14. Remove read and write permission on the `config` instance security object for the Example role.

```
setacl role:Example-rw config
CDISC0019I The access control list for the instance1 instance in the domain1 domain was
 updated.
```

15. View the permissions for `user1`. Note that the read and write permission on the `config` instance security object is now removed for `user1`.

```
lspermission user1 application-log:------
config:------
hosts:------
instance:------
jobgroup_default:------
jobs:------
jobs-override:------
system-log:------
```

16. To exit the interactive **`streamtool`** interface, enter **`exit`** or **`quit`**.

Related reference:

Streamtool commands

This reference section provides a description of each *streamtool* command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

9.1.2. Roles

A *role* is a collection of permissions that can be assigned to a user or group of users.

You can create roles in domains or instances.

Domain roles

By default, each domain has the following roles:

DomainAdministrator

This role has authority to administer the domain. For example, users and groups that are assigned this role can start and stop the domain, create and start instances, change and view permissions, and change the configuration of the domain. They also have authority to administer instances in the domain.

DomainUser

This role has authority to use the domain. For example, users and groups that are assigned this role can view domain properties, list the resources in the domain, and get information about the state of the domain.

Note

You cannot remove these default roles.

You can create more roles in your domain by using the **`streamtool mkdomainrole`** command. To assign these roles to groups or users, run the **`streamtool addgroupdomainrole`** or **`streamtool adduserdomainrole`** commands. Alternatively, you can use the Streams Console.

Instance roles

By default, each instance has the following roles:

InstanceAdministrator

This role has authority to administer the instance. For example, users and groups that are assigned this role can start and stop the instance, change and view permissions, change the configuration of the instance, and submit and cancel jobs.

InstanceUser

This role has authority to use the instance. For example, users and groups that are assigned this role can view instance properties, list the resources in the instance, and submit jobs. They can also perform the following tasks for jobs that they submitted: list the jobs, list the processing elements, cancel the jobs, and stop and restart processing elements.

Note

You cannot remove these default roles.

You can create more roles in your instances by using the **streamtool mkrole** command. To assign these roles to groups or users, run the **streamtool addgrouprole** or **streamtool adduserrole** commands. Alternatively, you can use the Streams Console.

You can use roles to secure domain and instance objects by setting the permissions for the role in the Streams access control list (ACL). For more information about granting permissions to roles, see "Viewing and configuring ACL settings for Streams domains and instances".

Related reference:

"Streamtool commands that require user authorization"

The **streamtool** command description provides information about the authority that is required to run each command.

streamtool mkrole command

The streamtool mkrole command creates a role in an instance.

streamtool mkdomainrole command

The streamtool mkdomainrole command creates a role in a domain.

9.1.3. Job groups

A *job group* is a group of jobs that have the same authority or permissions.

You can use job groups to limit who can perform tasks such as sending or receiving data from jobs, and stopping or restarting processing elements (PEs).

When you submit a job, you can specify which job group it belongs to. If you do not specify a job group, the submitted job uses the following job group: default.

To create more job groups, use the **streamtool mkjobgroup** command or the Streams Console. The job group name must satisfy the following requirements:

- The name must be unique among the job groups that are defined by you within the instance.
- The name must contain alphanumeric characters. You cannot use the following alphanumeric characters: ^!#$%&'*+,/;<>=?@[]` {|}~() You also cannot use the following Unicode and hexadecimal characters: u0000; u0001-u001F; u007F-u009F; ud800-uF8FF; uFFF0-uFFFF; x{10000}-x{10FFFF}.
- The maximum length for the name is 255 bytes.

The initial permissions for a new job group are set by the default: permissions for the jobs security object for an instance. The following example shows the `default:` permissions for the jobs security object for `instance1:`

```
$ streamtool getacl -d domain3 -i instance1 jobs
# object: jobs
# parent: instance
# owner: nobody
# persistent: yes
...
default:user:owner:rwsado
default:user:streamsadmin:rwsado
default:role:InstanceUser:---a--
default:role:DomainAdministrator:rwsado
default:role:InstanceAdministrator:rwsado
```

A new job group has the following permissions:

```
$ streamtool mkjobgroup -d domain3 -i instance1 -U jobuser groupA
CDISC0170I The following job group was created: groupA.
The job group was created for the instance1 instance in the domain3 domain.
$ streamtool getacl -d smd3 -i instance1 jobgroup_groupA
# object: jobgroup_groupA
# parent: jobs
# owner: joeuser
# persistent: yes
user:owner:rwsado
user:joeuser:rwsado
role:InstanceUser:---a--
role:DomainAdministrator:rwsado
role:InstanceAdministrator:rwsado
```

When you create a job group, you can set the initial permissions by changing the default: permissions on the jobs security object. The following example shows how to set the delete permission as a default for new job groups:

```
streamtool setacl -d smd3 -i instance1 default:role:InstanceUser+d jobs
CDISC0019I The access control list for the instance1 instance in the smd3 domain
 was updated.
```

You can set the permissions for job groups in the Streams access control list (ACL). For example, use the **`streamtool grantjobpermission`** and **`streamtool revokejobpermission`** commands to set the following permissions for users, groups, or roles:

ALL

This privilege is equivalent to having add, read, write, search, delete, and own authority for the jobgroup_*name* instance object.

DATAREAD

Receive data from the jobs in the job group. This privilege is equivalent to having read authority for the jobgroup_*name* instance object.

DATAWRITE

Send data to the jobs in the job group. This privilege is equivalent to having write authority for the jobgroup_*name* instance object.

JOBCONTROL

Submit new jobs or stop jobs in the job group. This privilege is equivalent to having add and delete authority for the jobgroup_*name* instance object.

OWN

Change the permissions of the jobs in the job group. This privilege is equivalent to having own authority for the jobgroup_*name* instance object.

SUBMIT

Submit new jobs in the job group. This privilege is equivalent to having add authority for the jobgroup_*name* instance object.

To allow jobs to import or export data, the administrator must grant permissions by modifying the job group. The following example shows how an administrator can change the permissions for the job

group to allow any jobs that are submitted by the user Alice to export data to any jobs created in the Streams instance:

```
streamtool grantjobpermission -d StreamsDomain -i StreamsInstance -J default
 user:Alice:DATAWRITE
```

The following example shows how an administrator can change the permissions for the job group to allow any jobs that are submitted by the user Bob to import data from any other jobs that are created in the Streams instance:

```
streamtool grantjobpermission -d StreamsDomain -i StreamsInstance -J default
 user:Bob:DATAREAD
```

Alternatively, you can use the **`streamtool setacl`** command to change the permissions for the jobgroup_default, or jobgroup_*name* security objects, where *name* is the name your job group. If you specify permissions that do not match the pre-defined job group permissions, they are designated as CUSTOM when you run the **`streamtool lsjobpermission`** command.

Related reference:

streamtool grantjobpermission command

The streamtool grantjobpermission command sets specific permissions for a job group.

streamtool mkjobgroup command

The streamtool mkjobgroup command creates a job group in an instance.

streamtool setacl command

The **`streamtool setacl`** command changes the access or default permissions for an object in a Streams instance.

9.1.4. Viewing and configuring ACL settings for Streams domains and instances

You can review and configure the access control list (ACL) settings for Streams domains and instances by using the **`streamtool`** command-line interface. You can also grant access privileges to domain and instance resources by using the Streams Console.

Procedure

- View ACL settings.
  - To view ACL settings for a domain, use the **`streamtool getdomainacl`** command.
  - To view ACL settings for an instance, use the **`streamtool getacl`** command.
- Configure ACL settings.
  - To configure ACL settings for a domain, use the **`streamtool setdomainacl`** command.
  - To configure ACL settings for an instance, use the **`streamtool setacl`** command.

Related tasks:

Administering a domain in the Streams Console

Related reference:

streamtool getdomainacl command

streamtool setdomainacl command

streamtool getacl command

streamtool setacl command

streamtool grantjobpermission command

streamtool revokejobpermission command

9.1.5. Configuring user access to Streams domains and instances

By default, Streams domains and instances are private and can be accessed only by the user who creates the domain or instance. To set up a domain or instance that can be accessed by multiple users, you can use either the Streams graphical user interfaces or **`streamtool`** commands.

Before you begin

If you are using PAM with a UNIX backend for user authentication, review the restrictions for domains and instances that are shared by multiple users.

About this task

In the following procedure, the optional administrator and user groups refer to a Linux or LDAP set of users. After you create the groups by using Linux or LDAP, you can add the groups to the Streams DomainAdministrator, DomainUser, InstanceAdministrator, and InstanceUser roles when you create the domain and instance. All members in the group have the DomainAdministrator, DomainUser, InstanceAdministrator, and InstanceUser permissions.

Restriction

To add the members of a Linux or LDAP administrator group and user group to the InstanceAdministrator and InstanceUser roles, you must create the instance by using the command-line procedure. This option is not supported in the Streams Console graphical user interface.

About this task

- Set up the domain or instance by using the Streams graphical user interfaces.
  - To set up a domain that is shared by multiple users:
    1. Create the domain. When you create a domain, Streams creates the DomainAdministrator and DomainUser roles, and adds the domain owner to the DomainAdministrator role.

       Optional: Specify a Linux or LDAP administrator group name and a user group name, which are added to the DomainAdministrator and DomainUser roles when the domain is created.

    2. Add users or groups to the DomainAdministrator and DomainUser roles by using the Streams Console.
  - To set up an instance that is shared by multiple users:

1. Create the instance by using the Streams Console. When you create an instance, Streams creates the InstanceAdministrator and InstanceUser roles, and adds the instance owner to the InstanceAdministrator role.

2. Add users or groups to the InstanceAdministrator and InstanceUser roles by using the Streams Console.

- Set up the domain or instance by using **`streamtool`** commands.
  - To set up a domain that is shared by multiple users:

    1. Create the domain by using the **`streamtool mkdomain`** command. When you create a domain, Streams creates the DomainAdministrator and DomainUser roles, and adds the domain owner to the DomainAdministrator role.

       Optional: Specify a Linux or LDAP administrator group name and a user group name, which are added to the DomainAdministrator and DomainUser roles when the domain is created.

       - To add an administrator group to the DomainAdministrator role, use the `--admin-grp` option.

       - To add a user group to the DomainUser role, use the `--user-grp` option.

    2. Add users or groups to the DomainAdministrator and DomainUser roles by using the **`streamtool adduserdomainrole`** and the **`streamtool addgroupdomainrole`** commands.

  - To set up an instance that is shared by multiple users:

    1. Create the instance by using the **`streamtool mkinstance`** command. When you create an instance, Streams creates the InstanceAdministrator and InstanceUser roles, and adds the instance owner to the InstanceAdministrator role.

       Optional: Specify a Linux or LDAP administrator group name and a user group name, which are added to the InstanceAdministrator and InstanceUser roles when you create the instance.

       - To add an administrator group to the InstanceAdministrator role, use the `--admin-grp` option.

       - To add a user group to the InstanceUser role, use the `--user-grp` option.

    2. Add users or groups to the InstanceAdministrator and InstanceUser roles by using the *streamtool adduserrole* and the *streamtool addgrouprole* commands.

Related concepts:

“Roles”

A *role* is a collection of permissions that can be assigned to a user or group of users.

Streams interfaces

See this information to learn more about the Streams Console.

Related reference:

Streamtool commands

This reference section provides a description of each **streamtool** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man command-name`.

Example 9.2. Configuring user access to an enterprise domain and instance by using roles

This example shows how you can use roles to configure user access to a Streams enterprise domain and instance. A *role* is a set of permissions or access rights. You can create roles or use default roles such as DomainAdministrator, DomainUser, InstanceAdministrator, and InstanceUser.

Before you begin

Create an enterprise domain. For instructions, see Chapter 8, “Setting up a Streams enterprise domain on multiple resources”.

About this task

Streams checks authority and manages permissions for all of the domain and instance objects at the domain level. Role information is stored in Apache ZooKeeper. You can assign roles to users or groups of users. You can then secure domain or instance objects by setting the permissions for the role in a Streams access control list (ACL).

Procedure

1. To complete this procedure in the interactive **streamtool** interface, enter the following command:

   ```
   streamtool
   ```

   Using the interactive **streamtool** interface saves you time. Streams caches some command options and information so that you do not have to reenter them. Also, you do not have to specify `streamtool` before each command. To exit the interactive **streamtool** interface, enter exit or quit.

2. Start the enterprise domain. You are prompted to provide an LDAP user ID and password, which must have the necessary authority to run the command. For example:

   ```
   startdomain -d testerdomain
   CDISA0064I Starting authorization service.
   User:tester
   Password:********
   CDISA0021I Starting domain testerdomain.
   ...
   ```

3. To view the roles in the domain, use the **streamtool lsdomainrole** command.

   When you create an enterprise domain, Streams creates the DomainAdministrator and DomainUser roles. By default, the domain owner is added to the DomainAdministrator role. For example:

   ```
   lsdomainrole
   Role: DomainAdministrator
   ```

```
  Users: tester
  Groups:
Role: DomainUser
  Users:
  Groups:
```

4. To assign a role to another user, use the **streamtool adduserdomainrole** command. For example:

```
adduserdomainrole DomainAdministrator frank
User:tester
Password:********
CDISC0150I Role DomainAdministrator for domain testerdomain has been assigned to user
 frank.
```

   Any user can log in to the domain but might not have permission to perform certain operations. In this case, the user frank is a domain administrator, and has the authority to access the testerdomain domain and run commands that require administrator authority, such as the following command:

```
lsdomainrole
User:frank
Password:********
Role: DomainAdministrator
  Users: frank,tester
  Groups:
Role: DomainUser
  Users:
  Groups:
```

5. To list all of the access control lists in the domain, users who have DomainAdministrator roles can use the **streamtool lsdomainacl** command. For example:

```
lsdomainacl -U tester
User tester password:********
# object: applications
# parent: domain
# owner: nobody
# persistent: yes
user:bsmith:--sa-o
role:DomainAdministrator:--sa-o
default:user:owner:rw--do
default:user:bsmith:rw--do
default:role:DomainAdministrator:rw--do
# object: charts
# parent: domain
# owner: nobody
# persistent: yes
user:bsmith:--sa-o
role:DomainAdministrator:--sa-o
default:user:owner:rw--do
default:user:bsmith:rw--do
default:role:DomainAdministrator:rw--do
# object: config
...
```

6. To assign a user to the DomainUser role, use the **streamtool adduserdomainrole** command. For example:

```
adduserdomainrole DomainUser wilriker
CDISC0150I Role DomainUser for domain testerdomain has been assigned to user wilriker.f
```

If a user with the DomainUser role tries to stop the domain, the command fails. For example:

```
stopdomain -d testerdomain -U wilriker
User wilriker password:********
CDISA5058E User wilriker is not authorized to stop domain testerdomain.
```

7. To list the permissions that a user has for objects in the domain, users with the DomainAdministrator role can use the **`streamtool lsdomainpermission`** command. For example, the user frank can list the permissions that the user `wilriker` has in the domain by entering the following command:

   ```
   lsdomainpermission wilriker -U frank
   User frank password:********
   applications:------
   charts:------
   config:------
   domain:--s---
   hosts:------
   instances:------
   system-log:------
   views:------
   ```

   The user `frank` can also list his own permissions for objects in the domain as shown in the following example:

   ```
   lsdomainpermission frank -U frank
   applications:--sa-o
   charts:--sa-o
   config:rw---o
   domain:rws-do
   hosts:rwsado
   instances:--sa-o
   system-log:rws--o
   views:--sa-o
   ```

8. To create an instance, use the **`streamtool mkinstance`** command. For example, the user frank can create an instance by using the following command:

   ```
   mkinstance -i testerinstance
   CDISA0026I Creating instance testerinstance in domain testerdomain.
   CDISA0027I Instance testerinstance created in domain testerdomain
    successfully.
   ```

9. To view the roles in the instance, use the **`streamtool lsrole`** command.

   When an instance is created Streams creates the InstanceAdministrator and InstanceUser roles. By default, the instance owner is added to the InstanceAdministrator role. For example:

   ```
   lsrole
   Role: InstanceAdministrator
     Users: frank
     Groups:
   Role: InstanceUser
     Users:
     Groups:
   ```

10. To list all of the access control lists in the instance, users that have InstanceAdministrator or DomainAdministrator roles can use the **`streamtool lsacl`** command. For example:

```
lsacl
# object: application-log
# parent: instance
# owner: nobody
# persistent: yes
user:frank:rws--o
user:bsmith:rws---
role:DomainAdministrator:rws--o
role:InstanceAdministrator:rws--o
# object: applications
# parent: instance
# owner: nobody
# persistent: yes
user:frank:--sa-o
role:DomainAdministrator:--sa-o
role:InstanceAdministrator:--sa-o
default:user:frank:rw--do
default:user:owner:rw--do
default:role:DomainAdministrator:rw--do
default:role:InstanceAdministrator:rw--do
# object: charts
...
```

11. To assign a role to another user, the instance administrator can use the **`streamtool adduserrole`** command. In the following example, the user frank adds the user `wilriker` to the `InstanceAdministrator` role:

    ```
    adduserrole -i testerinstance -d testerdomain InstanceAdministrator
     wilriker
    CDISC0156I Role InstanceAdministrator for instance testerinstance in
    domain testerdomain has been assigned to user wilriker.
    ```

12. To view the roles in the instance, use the **`streamtool lsrole`** command. For example:

    ```
    lsrole
    Role: InstanceAdministrator
      Users: frank,wilriker
      Groups:
    Role: InstanceUser
      Users:
      Groups:
    ```

13. To start the instance, an administrator can use the **`streamtool startinstance`** command. An administrator is a user who has either the DomainAdministrator or InstanceAdministrator role. Only an administrator can start or stop an instance. In the following example, the instance owner frank starts the instance by entering the following command:

    ```
    startinstance -i testerinstance -d testerdomain -U frank
    CDISA0030I Starting instance testerinstance in domain testerdomain.
    ...
    CDISA0031I Instance testerinstance started successfully in domain
     testerdomain.
    ```

    The user `wilriker`, who has the `InstanceAdministrator` role, can also start the instance with the following command:

    ```
    startinstance -i testerinstance -d testerdomain -U wilriker
    ```

```
CDISA0030I Starting instance testerinstance in domain testerdomain.
...
CDISA0031I Instance testerinstance started successfully in domain
 testerdomain.
```

The user `tester`, who has the `DomainAdministrator` role, can stop the instance with the following command:

```
stopinstance -i testerinstance -d testerdomain -U tester
User tester password:********
CDISA0032I Stopping instance testerinstance in domain testerdomain.
...
CDISA0033I Instance testerinstance stopped successfully in domain
 testerdomain.
```

Related concepts:

"Roles"

A *role* is a collection of permissions that can be assigned to a user or group of users.

9.1.6. Streamtool commands that require user authorization

The **`streamtool`** command description provides information about the authority that is required to run each command.

To run most commands, you must have specific permissions on Streams security objects. For more information about a command, enter `streamtool man` *`command-name`*.

By default, users or groups that are assigned to the DomainAdministrator, DomainUser, InstanceAdministrator, or InstanceUser roles have the permissions necessary to run a subset of **`streamtool`** commands. These commands are listed in Table 27. All other **`streamtool`** commands can be run by all authenticated users in Streams.

Notes:

- A member of the DomainAdministrator role is also authorized to run commands that are listed for the DomainUser, InstanceAdministrator, and InstanceUser roles.
- A member of the InstanceAdministrator role is also authorized to run the commands that are listed for the InstanceUser role.

Table 9.5. Streamtool commands that can be run by the default domain and instance roles

Role	Authorized streamtool commands	
DomainAdministrator	*adddomainhost*	*lsrole*
	addcertificate	*mkdomainrole*
	addgroupdomainrole	*mkhosttag*
	addgrouprole	*mkinstance*
	addhost	*mkjobgroup*
	adduserdomainrole	*mkrole*
	adduserrole	*mvdomainservice*

Role	Authorized streamtool commands	
	canceljob	*quiescehost*
	capturestate	*renewcertificate*
	checkdomainhosts	*requestcertificate*
	checkinstancehosts	*resetdomainmetrics*
	chhost	*restartdomainservice*
	chhosttag	*restartpe*
	chresourcespec	*restartservice*
	cleanlog	*resumehost*
	cpinstance	*revokejobpermission*
	displaype	*rmcertificate*
	exportkeystore	*rmdomain*
	exporttruststore	*rmdomainhost*
	gencertificate	*rmdomainproperty*
	getacl	*rmdomainrole*
	getdomainacl	*rmgroupdomainrole*
	getdomainproperty	*rmgrouprole*
	getdomainstate	*rmhost*
	getjobtopology	*rmhosttag*
	getlog	*rmjobgroup*
	getproperty	*rmproperty*
	getresourcestate	*rmresourcespec*
	grantjobpermission	*rmrole*
	importcertificate	*rmuserdomainrole*
	importkeystore	*rmuserrole*
	importtruststore	*setacl*
	lsacl	*setdomainacl*
	lsavailablehosts	*setdomainproperty*
	lscertificate	*setproperty*
	lsdomainacl	*startdomain*
	lsdomainpermission	*startinstance*
	lsdomainrole	*stopdomain*
	lshosts	*stopinstance*
	lshosttag	*stoppe*
	lsinstance	*submitjob*
	lsjobgroup	*updatecertificate*
	lsjobpermission	*updatepe*
	lsjobs	*updateusers*
	lspermission	*viewlog*

Role	Authorized streamtool commands	
	lspes	
DomainUser	*checkdomainhosts*	*getinstancecpu*
	getdomaincpu	*getresourcestate*
	getdomainmetrics	*lsavailablehosts*
	getdomainproperty	*lshosttag*
	getdomainstate	
InstanceAdministrator	*addgrouprole*	*mvservice*
	addhost	*resetinstancemetrics*
	adduserrole	*restartpe*
	canceljob	*restrictinstance*
	capturestate	*revokejobpermission*
	checkinstancehosts	*rmgrouprole*
	chresourcespec	*rmhost*
	cpinstancedisplayjob	*rminstance*
	displaype	*rmjobgroup*
	getacl	*rmproperty*
	getjobtopology	*rmresourcespec*
	getproperty	*rmrestrictedconfig*
	getrestrictedconfig	*rmrole*
	grantjobpermission	*rmuserrole*
	lsacl	*setacl*
	lshosts	*setproperty*
	lsjobgroup	*setrestrictedconfig*
	lsjobpermission	*startinstance*
	lsjobs	*stopinstance*
	lspermission	*stoppe*
	lspes	*submitjob*
	lsrole	*unrestrictinstance*
	mkjobgroup	*updateoperators*
	mkrole	*updatepe*
InstanceUser	*checkinstancehosts*	*lsjobs*
	displaype	*lspes*
	getjobtopology	*restartpe*
	getproperty	*submitjob*
	lshosts	*updatepe*
	lsjobgroup	

Related concepts:

"Roles"

A *role* is a collection of permissions that can be assigned to a user or group of users.

9.2. User authentication options for Streams

The default user authentication method for Streams domains is PAM or LDAP. For a basic domain, Streams uses PAM. For an enterprise domain, you can specify either LDAP or PAM as the default method when you create the domain, and then customize user authentication after the domain is created.

Related tasks:

Chapter 7, "Setting up a Streams basic domain on a single resource,"

A *basic domain* has a single Streams resource and user. This type of domain is typically used for test or development environments.

Chapter 8, "Setting up a Streams enterprise domain on multiple resources,"

An *enterprise domain* can have multiple resources and users. This type of domain is typically used for production environments. You can configure high availability to ensure that Streams can continue to run even if resources fail or are not available.

9.2.1. Order of user authentication checks by Streams

Your user authentication configuration determines how Streams authenticates users.

Streams authenticates users in the following order:

1. If configured, Streams attempts to authenticate by using the login module configuration. If the user cannot be authenticated, Streams continues to the next step.
2. If configured, Streams attempts to authenticate by using a client certificate. If the user cannot be authenticated, Streams continues to the next step.
3. If configured, Streams attempts to authenticate by using Kerberos. If the user cannot be authenticated, Streams continues to the next step.
4. Streams authenticates by using the default user authentication method that was specified when you created the domain (LDAP or PAM).

Related tasks:

"Setting up login module authentication for Streams users"

Streams authenticates users by using a Java Authentication and Authorization Service (JAAS). You can use the login modules that are included in the product, or you can create your own to customize the security settings of your domain. Using login modules is optional.

"Setting up client certificate authentication for Streams users" Use this procedure to set up client certificate authentication for a Streams domain by using X.509 certificates. Using client certificate authentication is optional.

"Setting up Kerberos authentication for Streams users" Use these procedures to set up Kerberos authentication for a Streams domain. Using Kerberos authentication is optional.

"Setting up the default user authentication method for a streams enterprise domain"

Before you create a Streams enterprise domain, use these guidelines to set up an LDAP server or the PAM authentication service for user authentication. After you create the domain, you can customize user authentication.

9.2.2. Customizing user authentication for a Streams enterprise domain

When you create an enterprise domain, you must specify either LDAP or PAM as the default user authentication method for Streams. After creating the domain, you can use login modules, client certificates, and Kerberos to customize user authentication.

Before you begin

Create an enterprise domain.

9.2.2.1. Setting up login module authentication for Streams users

Streams authenticates users by using a Java Authentication and Authorization Service (JAAS). You can use the login modules that are included in the product, or you can create your own to customize the security settings of your domain. Using login modules is optional.

About this task

To customize user authentication, you can use the **`streamtool setloginconfig`** command.

Procedure

- Customize user authentication by using the **`streamtool setloginconfig`** command. This command changes authentication configuration information for a domain. You must provide the information in a login configuration file.

  You can use the following commands to obtain information about the login modules in a domain:

  streamtool getloginconfig

  This command retrieves authentication configuration information for each login module that is used by the domain. You can optionally output the information to a file.

  streamtool lsloginmodule

  This command lists the login modules that are installed for a domain.

  For more information about these commands, enter `streamtool man` *`command-name`*.

Related tasks:

"Developing user-defined login modules for Streams"

Use this procedure and the sample login module that is installed with Streams to develop your own customized login module.

Related reference:

"Streams login modules"

Streams provides built-in login modules for customizing PAM, LDAP, X.509 certificate authentication, and Kerberos authentication.

"JAAS login configuration file example"

This example shows the format of the JAAS login configuration file and provides information about several file elements and attributes.

Streamtool commands

This reference section provides a description of each **`streamtool`** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

Streams login modules:

Streams provides built-in login modules for customizing PAM, LDAP, X.509 certificate authentication, and Kerberos authentication.

PAM authentication

Module class: com.ibm.streams.security.authc.module.PAMLoginModule

Module option:

service

The PAM authentication service name.

LDAP authentication

Module class: com.ibm.streams.security.authc.module.LDAPLoginModule

Module options:

serverUrl

LDAP server URL. This URL includes the host name and port number of the LDAP server, for example, ldap://ldap1.ibm.com:389.

userDnPattern

User DN Pattern. This pattern is used to create a distinguished name (DN) for a user during login, for example:

```
cn=*,ou=people,dc=ibm,dc=com, which is valid for any LDAP server type.
ADDOMAINNAME\\\*, which is valid for Windows Active Directory only.
```

When the user logs in, the user ID is substituted for the asterisk (*) in the pattern.

groupObjectClass

LDAP group object class that is used to search for group names.

groupSearchBaseDn

LDAP base DN that is used to search for groups.

groupAttributeWithUserNames

LDAP name of the element in the group record that contains the list of members in the group.

userAttributeStoredInGroupAttribute

LDAP name of the element in a user record that is stored in the group record.

userSecondaryLookup

LDAP user secondary lookup query that Streams uses to find the LDAP user name from the specified user ID, for example:

```
"(&(objectclass=ibmperson)(notesshortname=*)) uid".
```

X.509 certificate authentication

Module class: com.ibm.streams.security.authc.module.X509CertLoginModule

Module options: None

Kerberos authentication

Module class: com.ibm.streams.security.authc.module.KerberosLoginModule

Module options: None

Developing user-defined login modules for Streams:

Use this procedure and the sample login module that is installed with Streams to develop your own customized login module.

The Streams sample login module is located in the `$STREAMS_INSTALL/ samples/security/` directory.

Before you begin

The following software is required to create and package a customized login module:

- Apache Ant 1.9 or the latest stable version, which is available on the Apache Ant Project website.
- bnd 2.4.1.

Procedure

1. Implement a standard javax.security.auth.spi.LoginModule interface.

   For instructions, see the following Java Authentication and Authorization Service (JAAS) documentation on the Oracle Java Platform Standard Edition 8 Documentation website:

   - JAAS LoginModule Developer's Guide
   - JAAS Reference Guide

   Streams provides an SPI OSGi bundle (com.ibm.streams.security.authc.jar) for adding the following user and group principals that are specific to Streams into the authenticated subject when authentication is successful:

   - com.ibm.streams.security.authc.UserPrincipal
   - com.ibm.streams.security.authc.GroupPrincipal

   For more information, see the sample login module in the `$STREAMS_INSTALL/ samples/security/` directory.

2. Compile and package the implementation class files into an OSGi bundle.

   a. In the bnd `build.properties` file, update the **`bnd.path`** property and specify the absolute path where the bnd `.jar` file is located.

      Example:

      `bnd.path=/filepath/biz.aQute.bnd-2.4.1.jar`

   b. Create a `bnd.bnd` file and add standard OSGi manifest headers and the Java manifest header `Jaas-ModuleClass` with the fully qualified implementation class name.

      Example:

      ```
      Private-Package: com.ibm.streams.security.auth.module.xml.*
      Export-Package: com.ibm.streams.security.auth.module.xml
      Bundle-Activator: com.ibm.streams.security.auth.module.xml.Activator
      Bundle-Version: ${version-build}
      Jaas-ModuleClass:
       com.ibm.streams.security.auth.module.xml.SimpleUserLoginModule
      ```

3. Run the Streams **`streamsprofile.sh`** script by entering the following command:

   `source product-installation-root-directory/7.1.0.0/bin/streamsprofile.sh`

4. Run the Ant **`build.xml`** script by entering the following command:

   `ant`

   This command creates the OSGi .jar file for the customized login module in the build/bundles directory.

5. Configure Streams to use the login module.

   a. To install the login module, copy the OSGi .jar file into the login module directory that is specified on the **`security.loginModulePath`** domain property. The default directory is `%STREAMS_USER_HOME%/.streams/var/security/modules`.

   b. Update the JAAS login configuration file by using the **`streamtool setloginconfig`** command. For more information about this command, enter `streamtool man setloginconfig`.

Related reference:

"JAAS login configuration file example"

This example shows the format of the JAAS login configuration file and provides information about several file elements and attributes.

Streamtool commands

This reference section provides a description of each **`streamtool`** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man command-name`.

JAAS login configuration file example:

This example shows the format of the JAAS login configuration file and provides information about several file elements and attributes.

For more information about the file elements and attributes, see the notes that follow the example.

```
<?xml version="1.0" encoding="UTF-8"?>

<securityDomain xmlns="http://www.ibm.com/xmlns/prod/streams/security/domain/config/1.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
 <authentication>
   <jaas>
     <jaasConfig name="streams-jaas">
     <loginModule moduleClass="com.ibm.streams.security.authc.module.X509CertLoginModule" flag="sufficient">
     </loginModule>

     <loginModule moduleClass="com.ibm.streams.security.authc.module.PAMLoginModule" flag="sufficient">
      <moduleOption name="service" value="login" />
     </loginModule>

     <loginModule moduleClass="com.ibm.streams.security.authc.module.LDAPLoginModule" flag="sufficient">
     <moduleOption name="serverUrl" value="ldap://bluepages.ibm.com:389" />
     <moduleOption name="userDnPattern" value="uid=*,c=us,ou=bluepages,o=ibm.com" />
     <moduleOption name="userSecondaryLookup" value="(&amp;(objectclass=ibmperson)(notesshortname=*)) uid" />
     <moduleOption name="groupObjectclass" value="groupOfUniqueNames" />
     <moduleOption name="groupSearchBaseDn" value="ou=memberlist,ou=ibmgroups,o=ibm.com" />
     <moduleOption name="groupAttributeWithUserNames" value="uniquemember" />
     <moduleOption name="userAttributeStoredInGroupAttribute" value="dn" />
     </loginModule>
    </jaasConfig>
   </jaas>
  </authentication>
</securityDomain>
```

Notes:

- You can specify one or more `<loginModule>` elements under the `<jaasConfig>` element.
- Authentication occurs in the order that you list the login modules.
- The `moduleClass` attribute specifies the fully qualified implementation class name.
- The `flag` attribute controls the behavior as authentication proceeds down the list of modules, which is one of the following values:

  required:

  The login module is required to succeed. Authentication continues to proceed down the login module list even if the login module succeeds or fails.

  requisite:

  The login module is required to succeed. If it succeeds, authentication continues down the login module list. If it fails, authentication does not proceed down the login module list and control returns to the application.

  sufficient:

  The login module is not required to succeed. If it succeeds, authentication does not proceed down the login module list and control returns to the application. If it fails, authentication continues down the login module list.

  optional:

  The login module is not required to succeed. If it succeeds or fails, authentication continues to proceed down the login module list.

- Overall authentication succeeds if all required and requisite login modules succeed. If a `sufficient` login module is configured and succeeds, only the `required` and `requisite` login modules before that `sufficient` login module need to succeed for the overall

authentication to succeed. If no `required` or `requisite` login modules are configured for an application, at least one `sufficient` or `optional` login module must succeed.

Table 9.6. Login module configuration scenarios

Login module	Flag	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6	Scenario 7	Scenario 8
A	required	PASS	PASS	PASS	PASS	FAIL	FAIL	FAIL	
B	sufficient	PASS	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	PASS
C	requisite		PASS	PASS	FAIL		PASS	PASS	
D	optional		PASS	FAIL			PASS	FAIL	
Overall authentication		*PASS*	*PASS*	*PASS*	*FAIL*	*FAIL*	*FAIL*	*FAIL*	*PASS*

- Module options can be specified in the `<moduleOption>` element as a name pair value. You can specify zero or more `<moduleOption>` elements under the `<loginModule>` element.

9.2.2.2. Setting up client certificate authentication for Streams users

Use this procedure to set up client certificate authentication for a Streams domain by using X.509 certificates. Using client certificate authentication is optional.

Before you begin

In the following procedure, the **`openssl`** command is used to work with certificates. This command is included in the `openssl` package. To download this package, go to the OpenSSL website.

About this task

After you set up client certificate authentication, Streams attempts to use X.509 certificate authentication when it authenticates a user to the domain. If certificate authentication fails, Streams attempts to use Kerberos authentication. If Kerberos authentication fails, Streams uses the default authentication method for the domain, which is either LDAP or PAM.

Procedure

1. Obtain X.509 certificates.

   You can use certificates that are signed by a certificate authority (CA) or self-signed certificates. Streams supports certificates in Distinguished Encoding Rules (DER) format or Privacy Enhanced Mail (PEM) format.

   - If you are using certificates that are signed by a CA, complete the following steps:

     a. Obtain the following files from the CA:

        - Client certificate
        - CA certificate for the CA that issued the client certificate
        - Certificate Revocation List (CRL) of the CA that issued the client certificate

The process for requesting certificates depends on the issuing CA. Typically, you create a private key file and then create a certificate signing request (CSR) file that you send to the CA to sign. For more information, contact your CA.

If you receive a certificate in PEM format, remove any text that is outside of the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- markers. You can manually remove the text with an editor of your choice. This update ensures that only the base64 encoded data remains. The following example shows a client certificate that contains the correct text:

```
-----BEGIN CERTIFICATE-----
MIID+jCCA2OgAwIBAgICEA8wDQYJKoZIhvcNAQEFBQAwcTELMAkGA1UEBhMCVVMx
CzAJBgNVBAgTAk1OMQwwCgYDVQQKEwNJQk0xEDAOBgNVBAsTB1N0cmVhbXMxEzAR
Y29tMB4XDTE1MTAxNDE1NDkyN1oXDTE2MTAxMzE1NDkyN1owdzELMAkGA1UEBhMC
VVMxCzAJBgNVBAgTAk1OMQwwCgYDVQQKEwNJQk0xEDAOBgNVBAsTB1N0cmVhbXMx
FDASBgNVBAMTC3N0cmVhbXN1c2VyMSUwIwYJKoZIhvcNAQkBFhZzdHJlYW1zdXNl
ckB1cy5pYm0uY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCjaQlq1M5d
YQP6puRLOTqmZdKFFngpuFnoIF6I7uwMJ8jNZx0Y9rkxVTGCqvmxnwvADG0GV5UR
ilXFhTYJINo1ggo+Ohm819k7YVAESv34kryj+1f86yj00Tzq6ykefYMre7t8PIZo
QW8QvjLZEdbjZnMgXyOGHWz1rAt+0376XwIDAQABo4IBmTCCAZUwKQYJYIZIAYb4
QgENBBwWGk9wZW5TU0wgQ2xpZW50IENlcnRpZmljYXRlMEAGCCsGAQUFBwEBBDQw
MjAwBggrBgEFBQcwAYYkaHR0cDovL3Nob2UucmNoLnN0Z2xhYnMuaWJtLmNvbTo5
MDgwMIGWBgNVHSMEgY4wgYuAFBQ8NrJJ9ddVW4Z/LvDbtUvsC51UoXCkbjBsMQsw
CQYDVQQGEwJVUzELMAkGA1UECBMCTU4xEjAQBgNVBAcTCVJvY2hlc3RlcjEMMAoG
A1UEChMDSUJNMQwwCgYDVQQDEwNJQk0xIDAeBgkqhkiG9w0BCQEWEXNtc2hhb0B1
cy5pYm0uY29tggFYMAkGA1UdEwQCMAAwPgYDVR0fBDcwNTAzoDGgL4YtaHR0cDov
L3Nob2UucmNoLnN0Z2xhYnMuaWJtLmNvbS9jZXJ0cy9jcmwucGVtMA4GA1UdDwEB
/wQEAwIHgDATBgNVHSUEDDAKBggrBgEFBQcDAjAdBgNVHQ4EFgQUJqFG/AckkKoB
T8kRhyDCMNzmQ9gwDQYJKoZIhvcNAQEFBQADgYEAtqFIHeHakd1QAlxYRhKefqQg
VW0X7VRVJe7IhUKHBe8DXwSJmIETdxUbhzwNhmxijJkGyBu055YSlCz4X+wNaSC8
lDrO5g1ej3wh2cZFzVAYzyDhBo6urXiX9XEi94tc3/UBvwRHGP/MtmRJCbUyr5A7
p3Xu6ZBlpTyRZ18KRpU=
-----END CERTIFICATE-----
```

b. Verify the certificates by using the procedure in "Verifying certificates for Streams users".

To authenticate users, Streams interfaces can use a certificate or a password protected PKCS #12 file that contains the certificate. You can obtain the PKCS #12 file from a CA, or create this file by using the procedure in "Creating a password protected PKCS #12 file for certificates".

- If you are using self-signed certificates, complete the following steps:

  a. Generate an RSA private key by using the *openssl genpkey* command, for example:

  ```
  openssl genpkey -algorithm RSA -out selfsigned.key
  ```

  ## Restriction

  If the domain is configured to use Transport Layer Security (TLS) 1.2, you must include the `-pkeyopt rsa_keygen_bits:`*`numbits`* option with `numbits` set to 2048 or larger, for example:

  ```
  openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -out
   selfsigned.key
  ```

b. Generate a client certificate from the private key by using the **openssl req** command, for example:

```
openssl req -x509 -new -key selfsigned.key -out selfsigned.pem
-subj '/C=CA/ST=MN/O=IBM/OU=Streams/CN=selfsigned/'
```

Restriction:

If the domain is configured to use Transport Layer Security (TLS) 1.2, you must include the -sha256 option, for example:

```
openssl req -x509-sha256 -new -key selfsigned.key -out
 selfsigned.pem
-subj '/C=CA/ST=MN/O=IBM/OU=Streams/CN=selfsigned/'
```

To authenticate users, Streams interfaces can use a certificate or a password protected PKCS #12 file that contains the certificate. To create this file, use the procedure in "Creating a password protected PKCS #12 file for certificates".

2. Add the trusted certificate to the web management service truststore.

- If you use certificates that are signed by a CA, the trusted certificate is the certificate of the CA that issued the client certificate.

Attention:

If the certificate revocation status is valid, all client certificates that are issued by that CA can authenticate after you add the CA certificate to the web management service truststore. For more information about the certificate revocation status, see "Setting up client certificate revocation checking for Streams users".

- If you use self-signed certificates, the trusted certificate is the self-signed certificate.

To add the trusted certificate to the web management service truststore, enter the following Streams *streamtool* command:

```
streamtool addcertificate -d domainid --clientid trustedcert -f trustedcert.pem
```

A message similar to the following example is displayed:

```
Trusted client certificate for trustedcert imported successfully for
 domain domainid.
```

Notes:

- You can specify a PKCS #12 file that contains the trusted certificate on the *streamtool addcertificate* command. If specified, you are prompted for the PKCS #12 password. If the PKCS #12 file contains more than one trusted certificate, you are prompted to select a trusted certificate.

- You can enter any name for the clientid, but the preferred practice is to use a name that is associated with the trusted certificate. In the *streamtool* command example, the clientid of

trustedcert is associated with the file name of the trusted certificate. Another option is to use the subject CN of the trusted certificate, as shown in the following example:

```
openssl x509 -noout -subject -in trustedcert.pem
subject= /C=US/ST=MN/O=IBM/OU=Streams/CN=trustedcert/
emailAddress=admin@example.com
```

3. Set up Streams authorization for the certificate user.

   By default, the certificate user is the subject CN of the client certificate. You can display the client certificate subject information by entering the following **`openssl`** command:

   ```
   openssl x509 -noout -subject -in /streamscertificates/streamsuser.pem
   subject= /C=US/ST=MN/O=IBM/OU=Streams/CN=streamsuser/
   emailAddress=streamsuser@example.com
   ```

   In this example, the subject CN is streamsuser. For more information about the subject CN and other information in a client certificate, see "Setting the user ID pattern for certificate authentication".

   You can configure permissions for a certificate user by using roles or by setting access permissions for the user. Groups do not apply to certificate users. To configure permissions, see "Configuring user access to Streams domains and instances".

4. If you are using certificates that are signed by a CA, you might need to modify the default client revocation method that is used by Streams. For more information, see "Setting up client certificate revocation checking for Streams users". Client certificate revocation checking does not apply to self-signed certificates.

5. Enable client certificate authentication for the domain.

   The procedure to enable client certificate authentication depends on the Streams interface that you use.

Interface	Procedure

streamtool	Complete the following steps:
	
	Set the STREAMS_X509CERT environment variable to the path of the client certificate or a PKCS #12 file that contains the certificate.
	
	● Client certificate example
	
	`export STREAMS_X509CERT=/streamscertificates/streamsuser.pem`
	
	● PKCS #12 file example
	
	`export STREAMS_X509CERT=/streamscertificates/streamsuser.p12`
	
	Notes:
	
	● If you specify a PKCS #12 file, Streams prompts you for the PKCS #12 password. If the PKCS #12 file contains more than one certificate, Streams prompts you to select a user certificate.

Interface	Procedure
	• You can disable certificate authentication by running the **`unset STREAMS_X509CERT`** command.
Streams Studio	Complete the following steps: 1. Specify the path to the client certificate by using the Streams Explorer preference page. 2. Select the **`Use client certificate for authentication`** option when you add the domain connection. When you connect to the domain, Streams uses the certificate that you specified in the previous step for authentication. **Notes:** • If you specify a path to a PKCS #12 file that contains the certificate, Streams prompts you for the PKCS #12 password. This password is stored in Eclipse's Secure Storage. • If you specify a PKCS #12 file that contains more than one certificate, Streams prompts you to select a user certificate.
Streams Console	Complete the following steps: 1. Enable client authentication by setting the **`sws.clientAuthenticationEnabled`** domain property to true. You can use the **`streamtool setdomainproperty`** command to set this property. 2. For the changes to take effect, restart the domain. 3. Create a PKCS #12 file that contains the following files: • User certificate • User private key file • CA certificate, if the user certificate is signed by a CA For instructions, see “Creating a password protected PKCS #12 file for certificates”. 4. Import the PKCS #12 file into your browser. For additional information, see the help for your browser.
Streams for REST API	Complete the following steps: 1. Enable client authentication by setting the **`sws.clientAuthenticationEnabled`** domain property to true. You can use the **`streamtool setdomainproperty`** command to set this property. 2. For the changes to take effect, restart the domain.

Interface	Procedure
	3. Configure the Streams interface to authenticate using the client certificate. ● For the REST API, see Configuring security for the Streams REST API.

Related tasks:

"Setting up the default user authentication method for a Streams enterprise domain"

Before you create a Streams enterprise domain, use these guidelines to set up an LDAP server or the PAM authentication service for user authentication. After you create the domain, you can customize user authentication.

Setting preferences for Streams Studio

Use this information to set Streams Studio and Eclipse workbench preferences.

Adding domain connections for Streams Studio

To use Streams Studio with Streams, you must add at least one domain connection.

Related reference:

Streamtool commands

This reference section provides a description of each **streamtool** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

Related information:

Troubleshooting client certificate authentication problems

Verifying certificates for Streams users:

Use this procedure to verify that the X.509 certificates that you obtained from a certificate authority (CA) can be used to authenticate Streams users.

About this task

The following files are used in the procedure examples:

- `root-ca.pem`: Certificate of the CA that issued the `sub-ca.pem` file.
- `sub-ca.pem`: Certificate of the CA that issued the `user.pem` file.
- `user.pem`: Certificate of the user that was issued by the subordinate CA.
- `crl.pem`: Certificate revocation list that was generated by the subordinate CA.

Procedure

1. Display the text data in the client certificate, for example:

```
Certificate:
    Data:
```

```
        Version: 3 (0x2)
        Serial Number: 4111 (0x100f)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, ST=MN, O=IBM, OU=Streams, CN=subCA/
emailAddress=admin@example.com
        Validity
            Not Before: Oct 14 15:49:27 2015 GMT
            Not After : Oct 13 15:49:27 2016 GMT
        Subject: C=US, ST=MN, O=IBM, OU=Streams, CN=streamsuser/
emailAddress=streamsuser@example.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
              Modulus (1024 bit):
                00:a3:69:09:6a:d4:ce:5d:61:03:fa:a6:e4:4b:39:
                ec:0c:27:c8:cd:67:1d:18:f6:b9:31:55:31:82:aa:
                3c:ea:eb:29:1e:7d:83:2b:7b:bb:7c:3c:86:68:41:
                f9:b1:9f:0b:c0:0c:6d:06:57:95:11:8a:55:c5:85:
                3a:a6:65:d2:85:16:78:29:b8:59:e8:20:5e:88:ee:
                36:09:20:da:35:82:0a:3e:3a:19:bc:d7:d9:3b:61:
                50:04:4a:fd:f8:92:bc:a3:fb:57:fc:eb:28:f4:d1:
                6f:10:be:32:d9:11:d6:e3:66:73:20:5f:23:86:1d:
                6c:f5:ac:0b:7e:d3:7e:fa:5f
              Exponent: 65537 (0x10001)
        X509v3 extensions:
          Netscape Comment:
            OpenSSL Client Certificate
          Authority Information Access:
            OCSP - URI:http://example.com:9080

          X509v3 Authority Key Identifier:
            keyid:14:3C:36:B2:49:F5:D7:55:5B:86:7F:2E:F0:DB:B5:4B:EC:0B:9D:54
            DirName:/C=US/ST=MN/L=Rochester/O=IBM/CN=IBM/
emailAddress=admin@example.com
            serial:58

          X509v3 Basic Constraints:
            CA:FALSE
          X509v3 CRL Distribution Points:
            URI:http://example.com/crl.pem

          X509v3 Key Usage: critical
            Digital Signature
          X509v3 Extended Key Usage:
            TLS Web Client Authentication
          X509v3 Subject Key Identifier:
            26:A1:46:FC:07:24:90:AA:01:4F:C9:11:87:20:C2:30:DC:E6:43:D8
    Signature Algorithm: sha1WithRSAEncryption
        b6:a1:48:1d:e1:da:91:dd:50:02:5c:58:46:12:9e:7e:a4:20:
        09:b5:32:af:90:3b:a7:75:ee:e9:90:65:a5:3c:91:67:5f:0a:
        98:81:13:77:15:1b:87:3c:0d:86:6c:62:8c:99:06:c8:1b:b4:
        e7:96:12:94:2c:f8:5f:ec:0d:69:20:bc:94:3a:ce:e6:0d:5e:
        55:6d:17:ed:54:55:25:ee:c8:85:42:87:05:ef:03:5f:04:89:
        8f:7c:21:d9:c6:45:cd:50:18:cf:20:e1:06:8e:ae:ad:78:97:
        f5:71:22:f7:8b:5c:df:f5:01:bf:04:47:18:ff:cc:b6:64:49:
        46:95
```

2. Display the subordinate CA certificate subject and issuer data, for example:

   ```
   openssl x509 -noout -subject -issuer -in sub-ca.pem
   ```

   Output example:

```
subject= /C=US/ST=MN/O=IBM/OU=Streams/CN=subCA/emailAddress=admin@example.com
issuer= /C=US/ST=MN/L=Rochester/O=IBM/CN=rootCA/
emailAddress=admin@exmaple.com
```

3. Display the root CA certificate subject and issuer data, for example:

   ```
   openssl x509 -noout -subject -issuer -in root-ca.pem
   ```

   Output example:

   ```
   subject= /C=US/ST=MN/L=Rochester/O=IBM/CN=rootCA/
   emailAddress=admin@example.com
   issuer= /C=US/ST=MN/L=Rochester/O=IBM/CN=rootCA/
   emailAddress=admin@example.com
   ```

4. Verify the client certificate with the trusted certificates.

   a. Create a CA chain file, for example:

      ```
      cat sub-ca.pem root-ca.pem > ca-chain.pem
      ```

   b. Verify the client certificate with the trusted certificates, for example:

      ```
      openssl verify -CAfile ca-chain.pem user.pem
      ```

5. Verify the client revocation status with the trusted certificates, for example:

   ```
   openssl verify -crl_check -CAfile ca-chain.pem -CRLfile crl.pem user.pem
   ```

Creating a password protected PKCS #12 file for certificates:

Use this procedure to create a password protected PKCS #12 file that contains one or more certificates.

Before you begin

In the following procedure, the **`openssl`** command is used to work with certificates. This command is included in the `openssl` package. To download this package, go to the OpenSSL website.

About this task

The following files are used in the procedure examples:

- `root-ca.pem:` Certificate of the CA that issued the sub-ca.pem file.
- `sub-ca.pem:` Certificate of the CA that issued the user.pem file.
- `user.pem:` Certificate of the user that was issued by the subordinate CA.
- `user.key:` Private key of the user certificate.

Procedure

The following examples show how to create a password protected PKCS #12 file that contains one or more certificates. For more information about the *openssl pkcs12* command, enter man pkcs12.

- PKCS #12 file that contains one user certificate.

  ```
  openssl pkcs12 -export -in user.pem -caname user alias -nokeys -out user.p12 -passout pass:pkcs12
   password
  ```

- PKCS #12 file that contains one user certificate and its private key.

```
openssl pkcs12 -export -in user.pem -name user alias -inkey user.key -passin pass:key password -out
 user.p12 -passout pass:pkcs12 password
```

- PKCS #12 file that contains one CA certificate.

```
openssl pkcs12 -export -in sub-ca.pem -caname sub-ca alias -nokeys -out sub-ca.p12 -passout
 pass:pkcs12 password
```

- PKCS #12 file that contains a trusted CA chain of certificates.

```
cat sub-ca.pem root-ca.pem > ca-chain.pem
openssl pkcs12 -export -in ca-chain.pem -caname sub-ca alias -caname root-ca alias -nokeys -out ca-
chain.p12 -passout pass:pkcs12 password
```

- PKCS #12 file that contains a user certificate, user private key, and the associated CA certificate.

```
openssl pkcs12 -export -in user.pem -name user alias -inkey user.key -passin pass:key password -
certfile sub-ca.pem -caname sub-ca alias
-out user_and_sub-ca.p12 -passout pass:pkcs12 password
```

Setting up client certificate revocation checking for Streams users:

A certificate authority (CA) that issues an X.509 client certificate can also revoke that certificate. If the CA revokes a certificate, user authentication fails. The

`security.revocationMethod` domain property specifies the method that a Streams domain uses to check whether a certificate is revoked. If you are not using client certificate authentication, this property is ignored.

About this task

A CA must provide a method to check the revocation status of a client certificate. Two common certificate revocation methods for determining revocation status are the Certificate Revocation List (CRL) method and the Online Certificate Status Protocol (OCSP) method. Streams supports both methods.

You can use the default setting for the **`security.revocationMethod`** property or specify another value. The default value is automatic. This value specifies that the domain uses the following methods to check for revoked certificates:

- If the certificate contains an OCSP responder URL, the domain uses the OCSP method. For more information about this method, see the description of the ocsp value.
- If a CRL is referenced in the certificate or is specified on the **`security.revocationFile`** or **`security.revocationLdapUrl`** domain property, the domain uses the CRL method. For more information about this method, see the description of the crl value.
- If both OCSP and CRL information is provided, the domain uses the OCSP method first. If the OCSP responder does not reply, the CRL method is used.
- If no OCSP or CRL information is provided, certificate authentication fails.

To change the default value, you can set the **`security.revocationMethod`** property to one of the following values:

- `ocsp:` This value specifies that the domain uses OCSP information to check for revoked certificates. The OCSP information in the certificate must contain the URL of an OCSP responder. The OCSP responder determines the revocation status of the certificate. If the certificate does

not contain an OCSP responder URL or the OCSP responder does not respond, certificate authentication fails.

- `crl:` This value specifies that the domain uses CRL information to check for revoked certificates. The CRL is obtained from the location that is referenced in the certificate or specified on the **`security.revocationFile`** or **`security.revocationLdapUrl`** domain property. If you specify this value and do not provide a CRL, certificate authentication fails.
- `none:` This value specifies that no certificate revocation checks occur. The contents of the certificate and the **`security.revocationFile`** and **`security.revocationLdapUrl`** property settings are ignored.

Notes:

- If a client certificate does not include URLs for certificate revocation, you can use the **`security.revocationFile`** and **`security.revocationLdapUrl`** domain properties to refer to a CRL. In this case, add only the associated CA certificate to the web management service truststore for the domain because these properties apply to only one CA.
- If a client certificate does not include a URI to a CRL file, you might be able to manually download it from the CA. If you set the **`security.revocationFile`** property to the fully qualified path of the CRL file on the system, the CRL can be used during certificate authentication.

  Every Streams resource in the domain that is configured to run the authentication and authorization service must be able to access the CRL file. If you are not using a shared file system, there must be a copy of the CRL file on each resource.
- If the CRL is in an LDAP directory, set the **`security.revocationLdapUrl`** domain property using the URL obtained from the LDAP administrator.

  The format of the LDAP URL to a CRL is `ldap://host[:port]/dn?attribute`.

  - `host` is the domain name or IP address of the resource that is running the LDAP server.
  - `port` is the port number of the LDAP server, which is optional. The default port number is 389.
  - `dn` is the distinguished name of the object in the LDAP directory that contains the attribute.
  - `attribute` is named **`certificateRevocationList;binary`** and contains the CRL contents.

  You can update domain properties by using the **`streamtool setdomainproperty`** command. For more information about this command, enter `streamtool man setdomainproperty`.

  Important:

  If you use the **`streamtool`** command to update the **`security.revocationLdapUrl`** property, enclose the URL in quotation marks. Otherwise, the URL is not updated correctly.

For more information about domain properties, enter `streamtool man domainproperties`.

Related information:

Troubleshooting client certificate authentication problems

Setting the user ID pattern for certificate authentication:

Certificate authentication uses client authentication to authenticate the client connection, and then extracts information from the distinguished name (DN) of the client certificate to authenticate users. By default, Streams checks for a user ID in the DN common name (cn) field.

- If the user ID is valid and is authorized to access the Streams domain or instance, you can authenticate without having to enter a user ID and password.
- If the user ID is not valid, you are prompted for a user ID and password.

> **Note**
>
> This scenario applies to the Streams REST API only when you are accessing it from a browser and did not already log on by using the Streams Console in the same browser session.

You can use other DN field values to authenticate users by updating the pattern in the **`security.certificateUserRegularExpression`** domain property. This pattern uses a regular expression substitution, for example: ${element[, regex, replacement]}. This expression allows flexibility in constructing user IDs. For example, you can use the following patterns:

```
${element}
${element, regex, replacement}
```

About this task

The *user ID pattern* is a pattern that consists of reserved keywords and regular expressions. This pattern specifies the DN information that Streams uses to construct a user ID for certificate authentication.

If existing client certificates do not contain user information in any of the DN fields, you must create a new client certificate to use certificate authentication.

For more information about the *streamtool* commands in the following procedure, enter `streamtool man` *`command-name`*. For more information about the *security.certificateUserRegularExpression* property, enter streamtool man domainproperties.

Procedure

- To display the user ID pattern that is being used by Streams, enter the following command:

  ```
  streamtool getdomainproperty -d domain-id --zkconnect host:port
   security.certificateUserRegularExpression
  ```

  - The `-d` option specifies the domain identifier. If the STREAMS_DOMAIN_ID environment variable is set to this value, you do not need to specify the -d option on the command.
  - The `--zkconnect` option specifies the name of one or more host and port pairs for the configured external ZooKeeper ensemble. If you specify multiple host and port pairs, separate each pair with a comma. This value is the external ZooKeeper connection string. If the STREAMS_ZKCONNECT environment variable is set to this value, you do not need to specify the --zkconnect option on the command. To obtain this value, you can use the *streamtool getzk* command.

  The default command output shows the usage of the cn for the user name, for example:

```
security.certificateUserRegularExpression=${cn}
```

- To update the user ID pattern, enter the following command:

```
streamtool setdomainproperty -d domain-id --zkconnect host:port
 security.certificateUserRegularExpression=EMAILADDRESS
```

Example

This example shows how to update the default pattern to construct the user ID from DN field values other than the default cn field value.

The client certificate in this example contains the DN information in the following table. By default, ${cn} is used for the user ID pattern, which indicates that Streams uses RobertSmith as the user ID for certificate authentication.

Table 9.7. Streams Console example: DN information in the client certificate

User information	DN field	DN value
Common name of the certificate owner	cn	RobertSmith
Email address of the certificate owner	EMAILADDRESS	resmith@us.ibm.com
Organizational unit	OU	Streams
Organization	O	IBM
City	L	Raleigh
State	ST	NC
Country	C	US

The following pattern specifies that Streams constructs the user ID from the DN common name, city, and state values, and that different values are substituted for the city and state:

```
${cn}@${L,Raleigh,Rochester}@${ST,NC,MN}
```

To construct the user ID based on this pattern, Streams performs the following operations:

- Extracts the cn value (RobertSmith).
- Appends the L value (Raleigh) and replaces Raleigh with Rochester.
- Appends the ST value (NC) and replaces NC with MN.

The result is that Streams tests for a user ID of `RobertSmith@Rochester@MN`.

The following pattern specifies that Streams constructs the user ID from a portion of the DN email address value:

```
${EMAILADDRESS,(us.ibm.com),us}
```

To construct the user ID based on this pattern, Streams performs the following operations:

- Extracts the EMAILADDRESS value (resmith@us.ibm.com).
- Replaces us.ibm.com with us.

The result is that Streams tests for a user ID of resmith@us.

Related tasks:

"Setting up client certificate authentication for Streams users" Use this procedure to set up client certificate authentication for a Streams domain by using X.509 certificates. Using client certificate authentication is optional.

Related reference:

Streamtool commands

This reference section provides a description of each **streamtool** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

9.2.2.3. Setting up Kerberos authentication for Streams users

Use these procedures to set up Kerberos authentication for a Streams domain. Using Kerberos authentication is optional.

Introduction to Kerberos:

Kerberos is a network authentication protocol developed by the Massachusetts Institute of Technology (MIT). The Kerberos protocol uses secret-key cryptography to provide secure communications over a non-secure network. Primary benefits are strong encryption and single sign-on (SSO).

SSO allows users to access systems and services with one user ID and password. With Kerberos SSO, users are only prompted once for their user ID and password.

Kerberos runs as a third-party trusted server known as the Key Distribution Center (KDC). Each user and service on the network is a principal.

The KDC has three main components:

- An authentication server that performs the initial authentication and issues ticket-granting tickets for users.
- A ticket granting server that issues service tickets that are based on the initial ticket-granting tickets.
- A principals database of secret keys for all the users and services that it maintains.

Kerberos uses cryptographic tickets to avoid transmitting plain text passwords. User principals obtain ticket-granting tickets from the Kerberos KDC and present those tickets as their network credentials to gain access to Streams services and interfaces.

Kerberos shares a secret key with the KDC. This secret key is known only to the KDC and the service principal on each Streams resource. The service principal for Streams is the authentication and authorization service. The authentication and authorization service on each resource must be registered with the KDC. The Kerberos administrator generates a keytab file that the authentication and authorization service uses to authenticate to the KDC.

Related information:

Kerberos documentation

Prerequisites for using Kerberos authentication with Streams:

Ensure that you satisfy these requirements for using Kerberos with Streams.

- Supported Kerberos V5 implementations

Streams supports the following Kerberos V5 implementations:

- MIT Kerberos V5 Release 1.14 for Linux
- Microsoft Active Directory, Windows Server 2016

Other V5 implementations might work with Streams but are not tested. To install and configure Kerberos, see the Kerberos documentation.

Restrictions for Kerberos realm, service principal, and user principal names

Ensure that you satisfy any restrictions for Kerberos realm, service principal, and user principal names.

The following restrictions are important:

- Use ASCII characters only in service principal and user principal names.
- Do not use control characters in realm, service principal, and user principal names.

For the most complete and current information about name restrictions, see the Kerberos documentation.

RPM requirements

To use Kerberos with Streams, you must install the following client RPM packages on all resources that run the Streams authentication and authorization service.

- Required RPMs for RHEL 6 and 7
  - krb5-libs
  - krb5-workstation

Java requirement

Kerberos support requires the JRE that is shipped with Streams. This JRE is installed in the following directory:

product-installation-root-directory`/7.1.0.0/java`

Configuring Kerberos authentication for Streams users:

Use this procedure to configure Kerberos authentication and single sign-on (SSO) for Streams.

Before you begin

Ensure that you satisfy the requirements in "Prerequisites for using Kerberos authentication with Streams". ´

About this task

The following Kerberos terms are used in this procedure:

- *Key Distribution Center (KDC)*: The Kerberos KDC is the trusted third party that manages user authentication. For more information, see "Introduction to Kerberos".
- *service principal*: The Kerberos service principal is the Streams authentication and authorization service.

- *user principal*: User principals are Kerberos users who access Streams services and interfaces.

This procedure includes the following main tasks:

- Set up the Streams authentication and authorization service as the Kerberos service principal.
- Update the Streams security properties that are used by Kerberos. You can accept the default values for most of these properties.
- Obtain and cache the Kerberos ticket-granting tickets that are used to authenticate user principals.
- Enable Kerberos authentication for the Streams user interfaces.

After you set up Kerberos authentication, Streams attempts to use Kerberos ticket-granting tickets for user authentication. If Kerberos authentication fails, Streams uses the default authentication method for the domain, which is either LDAP or PAM.

Procedure

1. On all resources that are running the Streams authentication and authorization service, set up the service as a service principal for Kerberos.

   The following values are used in the examples:

   - streams-aas: The Kerberos service principal name for the authentication and authorization service.
   - domain1.ibm.com: The fully qualified name of the Streams resource that is running the authentication and authorization service.
   - IBM.COM: The name of the Kerberos realm.

   Complete the following steps on all resources that are running the authentication and authorization service:

   a. Create a service principal for the authentication and authorization service, as shown in the following examples.

      - MIT Kerberos example:

        ```
        kadmin: addprinc -randkey <streams-aas/domain1.ibm.com@IBM.COM>
        ```

      - Microsoft Active Directory example:

        - Create a user in Active Directory for the Kerberos service principal, for example: streams-aas.
        - Enable encryption in the user account, for example: AES256-SHA1.
        - Create a service principal, for example:

          ```
          setspn -A streams-aas/domain1.ibm.com streams-aas
          ```

   b. Create a .keytab file for the authentication and authorization service as shown in the following examples. The keytab encryption keys must be compatible with the supported user encryption types.

      - MIT Kerberos example:

```
kadmin: ktadd -norandkey -k streams-aas.keytab streams-aas/
domain1.ibm.com@IBM.COM
```

- Microsoft Active Directory example:

```
ktpass -out .\streams-aas.keytab -princ streams-aas/
domain1.ibm.com@IBM.COM
-mapUser streams\streams-aas
-mapOp set
-pass password
-crypto AES256-SHA1
-pType KRB5_NT_PRINCIPAL
```

c. Copy the .keytab file that you created in the previous step to the following directory on the resource that is running the authentication and authorization service:

user-home-directory/.streams/var/security/keytabs/streams-aas.keytab

---

Notes:

- The user and group owner of the .keytab file is the *user:group* that runs the authentication and authorization service on the resource. For example, if your username is streamsadmin and your group is streamsgroup, enter the following command:

  ```
  chown streamsadmin:streamsgroup streams-aas.keytab
  ```

- Because the .keytab file contains encrypted passwords, anyone with read permission on the file can run a command in the Kerberos realm for the principals in the realm. Restrict and monitor permissions on this file so that only the principal whose credentials the application is authenticating with has read access to the file. For example:

  ```
  chmod 400 streams-aas.keytab
  ```

- You can configure permissions for a Kerberos user by using roles or by setting access permissions for the user. Groups do not apply to Kerberos users. To configure permissions, see "Configuring user access to Streams domains and instances".

---

d. Verify that you are able to log in using the .keytab file by using the **kinit** program that is installed with Streams, for example:

```
$STREAMS_INSTALL/java/jre/bin/kinit -f -k -t ./streams-aas.keytab streams-aas/
domain1.ibm.com@IBM.COM
```

If your krb5.conf file is not found in the default location of /etc/krb5.conf, you can use the following alternative command:

```
$STREAMS_INSTALL/java/bin/java -Djava.security.krb5.conf=path-to-krb5.conf
 com.ibm.security.krb5.internal.tools.Kinit -f -k -t ./streams-aas.keytab streams-aas/
domain1.ibm.com@IBM.COM
```

After the .keytab file is verified, remove the credential cache file in the following directory:

```
user-home-directory/krb5cc_userID
```

2. Update Streams security properties that are used for Kerberos.

Complete the following steps on all resources that run the authentication and authorization service:

a. Enable Kerberos by setting the *security.kerberosEnabled* property to true. By default, this property is set to false, and Kerberos is not enabled. You can set this property by using the **`streamtool setdomainproperty`** command, for example:

```
streamtool setdomainproperty security.kerberosEnabled=true
```

b. If necessary, update the default settings for the following Streams properties for Kerberos.

You can update the following property settings by using the **`streamtool setdomainproperty`** command. For more information about each property, use the **`streamtool man domainproperties`** command.

aas.kerberosServicePrincipal

Specifies the service principal name for the Streams authentication and authorization service. This name is registered with the Kerberos KDC.

Example:

```
streamtool setdomainproperty aas.kerberosServicePrincipal=streams-aas/%STREAMS_RESOURCE
%@IBM.COM (aas.kerberosServicePrincipal%3Dstreams-aas/%25STREAMS_RESOURCE%25@IBM.COM)
```

aas.kerberosKeytabFile

Specifies the location of the Kerberos .keytab file for the Streams authentication and authorization service.

Example:

```
streamtool setdomainproperty aas.kerberosKeytabFile=%STREAMS_USER_HOME%/.streams/var/
security/keytabs/streams-aas.keytab
```

security.kerberosConfigurationFile

Specifies the location of the Kerberos configuration file. This file contains essential information such as the default KDC, default realm, and other settings that are used by Streams. For an example of this file, see "Sample Kerberos configuration file".

Example:

```
streamtool setdomainproperty security.kerberosConfigurationFile=/etc/krb5.conf
```

security.kerberosKDC

Specifies the default Kerberos KDC.

- This property overrides the default KDC value on the **`security.kerberosConfigurationFile`** property.
- If this property is set, you must also set the **`security.kerberosRealm`** property.

security.kerberosRealm

Specifies the default Kerberos realm.

- This property overrides the default realm value on the **`security.kerberosConfigurationFile`** property.

- If this property is set, you must also set the **`security.kerberosKDC`** property.

security.kerberosDebug

Specifies to turn Kerberos debugging for diagnostics on or off. By default, this property is set to `false`, and debugging is turned off. To turn on debugging, set to `true`.

c. Optional: Update the default rules for mapping Kerberos principal names to Streams user names.

d. Restart the Streams domain for the configuration changes to take effect.

3. Obtain and cache Kerberos ticket-granting tickets. These tickets are used by Streams to authenticate users with the authentication and authorization service.

Notes:

- The JRE does not support the ticket cache that is generated by the MIT Kerberos **`kinit`** command. You must use the **`kinit`** program that is installed with Streams to obtain and cache Kerberos ticket-granting tickets for Streams users.
- Before running the **`kinit`** command, a user must be registered as a user principal with the Kerberos KDC.

To obtain and cache a Kerberos ticket-granting ticket, enter the following command:

```
$STREAMS_INSTALL/java/jre/bin/kinit -f
```

To list the ticket-granting tickets in the ticket cache, enter the following command:

```
$STREAMS_INSTALL/java/jre/bin/klist
```

4. Enable Kerberos authentication for the Streams user interfaces.

The procedure to enable Kerberos authentication depends on the Streams interface that you use.

Interface	Procedure
streamtool	When you updated the **`security.kerberosEnabled`** property in Step 3a, Kerberos authentication was enabled for the **`streamtool`** command-line interface.
Streams Console	For instructions, see "Configuring the Streams Console to use Kerberos authentication".
Streams Studio	For instructions, see "Enabling Streams Studio for Kerberos authentication".
Streams for REST API	Kerberos authentication is not supported.

Related tasks:

"Managing the expiration of Kerberos client tickets for Streams users"

Streams requires that you use the JRE *kinit* program to initialize the Kerberos ticket cache. When you configure Kerberos for Streams by using this program, client tickets are not automatically renewed.

Updating the default rules for mapping Kerberos principal names to Streams user names:

You can use the default setting for the **`security.kerberosPrincipalMapping`** property to map Kerberos names to Streams names, or specify your own mapping rules.

If you use the default setting, the first component of the Kerberos user principal name is used as the mapped name for the default Kerberos realm. For example, if the default realm is IBM.COM, the Kerberos user principal `streamsuser1@IBM.COM` is mapped to Streams user `streamsuser1`.

Procedure

To specify your own mapping rules, you can use the **`streamtool`** command-line interface.

- If you use the **`streamtool`** command-line interface, enter the following command:

  `streamtool setkrbprincmapping --file` *`path-to-mapping-file`*

  This command sets the **`security.kerberosPrincipalMapping`** property to the mapping in the specified mapping file. For more information about this command, enter `streamtool man setkrbprincmapping`.

The mapping file contains one or more mapping rules in the format RULE:expression or DEFAULT, for the default rule.

- Each rule must be on a separate line. The first rule in the file is processed first. Processing continues down the list until a match is found. If no match is found, the mapping of the principal name fails.
- The expression is in the following format:

  `[n:string](regexp)s/pattern/replacement/`

  - [n:string] indicates a matching rule where n declares the number of components in the principal, excluding the realm component. If this rule matches, a value is formed by the components of the principal name in the order that is specified in the string.
  - The following variables are used:
    - $0 indicates the realm component of the principal name.
    - $1 indicates the first component of the principal name.
    - $2 indicates the second component of the principal name.

    For example, if the input principal name is `streamsadmin/admin@IBM.COM`, the first component ($1) is streamsadmin, the second component ($2) is admin, and the realm component ($0) is IBM.COM.
  - If the value produced by `[n:string]` matches the regular expression in `(regexp)`, the `s/pattern/replacement/` substitution is invoked. The pattern in the substitution expression is a regular expression that is used to locate the portion of the string to replace, and replacement is the value to use for replacing the matched section.

Example 9.3. Examples:

In the following examples, IBM.COM is the default Kerberos realm.

- To map all principals in the default realm to their first component, use the default setting. The mapping file contains the following line:

  ```
  DEFAULT
  ```

  Result:

  - `streamsuser@IBM.COM` maps to `streamsuser`.
  - `streamsadmin/admin@IBM.COM` maps to `streamsadmin`.

- To map all principals in the default realm with the second component /admin to admin and all other principals to their first component, include the following lines in the mapping file:

  ```
  RULE:[2:$1%$2@$0](.*%admin@IBM.COM)s/.*/admin/ DEFAULT
  ```

  Result:

  - `streamsadmin/admin@IBM.COM` maps to `admin`.
  - `streamsuser@IBM.COM` maps to `streamsuser`.

- To map all principals in the STREAMS.COM realm to their first component and all principals in the default realm to their first component, include the following lines in the mapping file:

  ```
  RULE:[1:$1@$0](.*@STREAMS.COM)s/@STREAMS.COM// DEFAULT
  ```

  Result:

  - `user1@STREAMS.COM` maps to `user1`.
  - `streamsuser1@IBM.COM` maps to `streamsuser1`.

For more information about the format for rule expressions, see the description of the **`security.kerberosPrincipalMapping`** domain property. To access domain property descriptions, enter **`streamtool man domainproperties`**.

You can use the **`streamtool getkrbprincmapping`** command to retrieve the current Kerberos principal mapping for a specified domain. For more information about this command, enter *`streamtool man getkrbprincmapping`*.

Configuring the Streams Console to use Kerberos authentication:

Use this procedure to configure Kerberos authentication for Streams Console users.

Before you begin

Configure Kerberos for Streams.

Procedure

1. Configure the Mozilla Firefox browser on Linux or the Microsoft Internet Explorer browser on Windows.

   ## Restriction:

   The Google Chrome browser is not supported.

The Streams Console uses the Simple and Protected GSS-API Negotiation (SPNEGO) protocol to authenticate users with Kerberos credentials. To use Kerberos authentication, you must configure your browser to use SPNEGO.

- To configure the Firefox browser on Linux to use SPNEGO, complete the following steps:

  a. Start Firefox.

  b. In the address field, type **`about:config`** and press **Enter**.

  c. In the Search field, type **`network.n`** and press **Enter**.

  d. Double-click **`network.negotiate-auth.trusted-uris`**, enter the host name of the web management service, and click **OK**. For example, if the URL for accessing the Streams Console is https://streamshost.ibm.com:8443/ streams/domain/console, you enter https://streamshost.ibm.com in the **`network.negotiate-auth.trusted-uris`** field.

  e. Double-click **`network.negotiate-auth.delegation-uris`**, enter the host name of the web management service, and click **OK**. This is the same value that you entered in the previous step.

  f. Restart your Firefox browser to activate this configuration.

- To configure the Internet Explorer browser on Windows to use SPNEGO, complete the following steps:

  a. Start Internet Explorer.

  b. Click **Tools** > **Internet Options** > **Security**.

  c. Select the Trusted Site icon and click **Sites**. In the Trusted sites window, add the host name, for example: https://streamshost.ibm.com.

  d. Select the **Local intranet** icon and click **Sites**.

  e. In the Local intranet window, complete the following steps:

     - Select the **Include all local (intranet) sites not listed in other zones** check box and click **Advanced**.

     - In the **Add this web site to the zone** field, enter the host name of the web management service. For example, if the URL for accessing the Streams Console is `https://streamshost.ibm.com:8443/streams/ domain/console`, you enter `https://streamshost.ibm.com` in this field.

     - Click **Close** and then **OK** to return to the Internet Options window.

  f. In the Internet Options window, complete the following steps:

     - Click the **Advanced** tab and scroll to the **Security** settings.

     - Ensure that the **Enable Integrated Windows Authentication** check box is selected.

     - Click **OK**.

g. Restart Internet Explorer to activate this configuration.

2. On all resources that are running the Streams web management service, set up the service as a service principal for Kerberos.

   The following values are used in the examples:

   - `HTTP:` The Kerberos service principal name for the web management service.

     This is the name that browsers use by default.

   - `domain1.ibm.com:` The fully qualified name of the Streams resource that is running the web management service.

   - `IBM.COM:` The name of the Kerberos realm.

   Complete the following steps on all resources that are running the web management service:

   a. Create a service principal for the web management service as shown in the following examples.

      - MIT Kerberos example:

        ```
        kadmin: addprinc -randkey HTTP/domain1.ibm.com@IBM.COM
        ```

      - Microsoft Active Directory example:

        ○ Create a user in Active Directory for the Kerberos service principal, for example: HTTP.

        ○ Enable encryption in the user account, for example: `AES256-SHA1`.

        ○ Create a service principal, for example:

        ```
        setspn -A HTTP/domain1.ibm.com HTTP
        ```

   b. Create a .keytab file for the web management service as shown in the following examples. The keytab encryption keys must be compatible with the supported user encryption types.

      - MIT Kerberos example:

        ```
        kadmin: ktadd -norandkey -k streams-sws.keytab HTTP/
        domain1.ibm.com@IBM.COM
        ```

      - Microsoft Active Directory example:

        ```
        ktpass -out .\streams-sws.keytab -princ HTTP/domain1.ibm.com@IBM.COM
        -mapUser streams\HTTP
        -mapOp set
        -pass password
        -crypto AES256-SHA1
        -pType KRB5_NT_PRINCIPAL
        ```

   c. Copy the .keytab file that you created in the previous step to the following directory on the resource that is running the web management service:

      user-home-directory`/.streams/var/security/keytabs/streams-sws.keytab`

Notes:

- The user and group owner of the .keytab file is the **`user:group`** that runs the web management service on the resource. For example, if your username is `streamsadmin` and your group is `streamsgroup`, enter the following command:

  ```
  chown streamsadmin:streamsgroup streams-sws.keytab
  ```

- Because the .keytab file contains encrypted passwords, anyone with read permission on the file can run a command in the Kerberos realm for the principals in the realm. Restrict and monitor permissions on this file so that only the principal whose credentials the application is authenticating with has read access to the file. For example:

  ```
  chmod 400 streams-sws.keytab
  ```

- You can configure permissions for a Kerberos user by using roles or by setting access permissions for the user. Groups do not apply to Kerberos users. To configure permissions, see "Configuring user access to Streams domains and instances".

d. Verify that you are able to log in using the .keytab file.

Important

You must verify the .keytab file by using the MIT Kerberos **`kinit`** program, not the **`kinit`** program that is installed with Streams.

Example:

```
/usr/bin/kinit -f -k -t ⌂/.streams/var/security/keytabs/streams-sws.keytab HTTP/
domain1.ibm.com@IBM.COM
```

If your krb5.conf file is not found in the default location of /etc/krb5.conf, set the KRB5_CONFIG environment variable to the correct path and then run the *kinit* command.

After the .keytab file is verified, remove the credential cache file by running the following command:

```
/usr/bin/kdestroy
```

e. Repeat Steps 2(a-d) on all resources that are running the web management service.

3. Update Streams security properties that are used for Kerberos.

   You can update the property settings by using the **`streamtool setdomainproperty`** command. For more information about each property, use the *`streamtool man domainproperties`* command.

   a. Update the **`sws.kerberosServicePrincipal`** property, if needed.

      This property specifies the service principal name for the Streams web management service. This name is registered with the Kerberos KDC.

      The default value for this property is `HTTP/%STREAMS_RESOURCE%`, where `%STREAMS_RESOURCE%` resolves to the host name for the web management service. For example, if the URL for accessing the Streams Console is

`https://streamshost.ibm.com:8443/streams/domain/console`, the host name for the web management service is streamshost.ibm.com in this field.

b. Update the **`sws.kerberosKeytabFile`** property on all resources that run the Streams web management service.

This property specifies the location of the Kerberos .keytab file for the web management service.

Example:

```
streamtool setdomainproperty sws.kerberosKeytabFile=%STREAMS_USER_HOME%/.streams/var/security/
keytabs/streams-sws.keytab
```

4. Obtain and cache Kerberos ticket-granting tickets. These tickets are used by Streams to authenticate users with the web management service.

Note: Before running the *kinit* command, a user must be registered as a user principal with the Kerberos KDC.

To obtain and cache a Kerberos ticket-granting ticket, enter the following command:

```
/usr/bin/kinit -f
```

To list the ticket-granting tickets in the ticket cache, enter the following command:

```
/usr/bin/klist
```

Enabling Streams Studio for Kerberos authentication:

Enable Kerberos authentication for Streams Studio users.

Restriction:

Streams Studio supports Kerberos authentication on Linux only.

Before you begin

Configure Kerberos for Streams.

Procedure

Enable Kerberos authentication for Streams Studio on Linux by selecting the **Use Kerberos authentication** option when you add the domain connection.

When you connect to the domain, Streams uses Kerberos to authenticate Streams Studio users.

Related tasks:

Adding domain connections for Streams Studio

To use Streams Studio with Streams, you must add at least one domain connection.

Managing the expiration of Kerberos client tickets for Streams users:

Streams requires that you use the IBM JRE **`kinit`** program to initialize the Kerberos ticket cache. When you configure Kerberos for Streams by using this program, client tickets are not automatically renewed.

Procedure

You can use the following options to manage the expiration of Kerberos client tickets.

- Automatically run the **`kinit -k`** command by setting up a cron job on the Linux system. This approach avoids ticket expiration issues.
- Obtain a new ticket by running the **`kinit -k`** command. The command must be run on a regular basis to ensure that it runs before the ticket expires.

Command example:

```
$STREAMS_INSTALL/java/jre/bin/kinit -f -k -t path-to-keytab-file user-principal
```

Sample Kerberos configuration file:

The Kerberos configuration file, `krb5.conf`, contains essential information such as the default Kerberos Key Distribution Center (KDC), default realm, and other settings that are used by Streams.

When you configure Kerberos for Streams, you specify the location of the `krb5.conf` file on the Streams **`security.kerberosConfigurationFile`** property.

```
================sample krb5.conf ==============
[logging]
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log

[libdefaults]
default_realm = IBM.COM
dns_lookup_realm = false
dns_lookup_kdc = false
ticket_lifetime = 24h
renew_lifetime = 7d
forwardable = true

[realms]
IBM.COM = {
kdc = server1.ibm.com
admin_server = server1.ibm.com
}

[domain_realm]
.ibm.com = IBM.COM
ibm.com = IBM.COM
```

Related tasks:

“Configuring Kerberos authentication for Streams users”

Use this procedure to configure Kerberos authentication and single sign-on (SSO) for Streams.

9.2.3. Generating authentication keys for Streams

Use this procedure to generate public and private keys for Streams. Generating public and private keys eliminates the need for Streams users to enter a password when they run **`streamtool`** commands.

Before you begin

To successfully use generated authentication keys when a domain is running in an environment without a shared file system, the **`streamtool genkey`** command must be run on each resource where the user runs **`streamtool`** commands.

Procedure

Generate public and private keys.

- To create public and private keys for the user who is entering the command, enter streamtool genkey.
- If LDAP is used to authenticate Streams users, you can create public and private keys for another user by entering `streamtool genkey -U` *`user-id`*.

The **`streamtool genkey`** command generates the following files:

- Private key: *`user-id`*`_priv.pem`
- Public key: *`user-id`*`.pem`

Streams stores the private key in the *`user-home-directory/`*`.streams/key/`*`domain-id` directory. The public key is stored in ZooKeeper.

Related reference:

streamtool genkey command

The streamtool genkey command generates a public and private key pair.

9.2.4. Streams security session timeout property

The security session timeout is the number of seconds before a Streams user session expires. This property setting is independent of the PAM or LDAP authentication service.

You can configure the **`security.sessionTimeout`** property when you create a Streams domain by using the **`streamtool mkdomain`** command. The default value is 14400 seconds (4 hours).

To update this property for an existing Streams domain, use the **`streamtool setdomainproperty`** command.

Related reference:

streamtool mkdomain command

streamtool setdomainproperty command

9.3. Linux users and Streams jobs

To determine which Linux user has the authority to run the jobs for a Streams instance, you must first determine which user is running the domain controller services for your domain. The security implications of the **`instance.runAsUser`** and **`instance.canSetPeOSCapabilities`** property settings also need to be considered.

Identifying which Linux user runs Streams jobs

- When you use SSH to start domain controller services, they run as the user that starts the domain.
- When resources are individually installed and domain controller services are running as system services, the domain controller services run as the user that was specified when registering the domain controller services as system services.

Note

A domain might contain a mixture of resources, for example, resources with domain controller services started by using SSH and resources with domain controller services started as system services.

Understanding user authority to run Streams jobs

If the **`instance.runAsUser`** property is not specified, the instance services and PE processes run as the domain controller service user. If this occurs, PE processes on different systems might be running as different users even for the same job.

If **`instance.runAsUser=`***operating_system_user* is specified, instance services and PE processes run as the *operating_system_user* on all instance resources. If **`instance.canSetPeOSCapabilities=true`** is specified, the PE processes in the Streams instance can run with special operating system capabilities. These settings are only valid when domain controller services are running as system services.

Important

Using the **`instance.runAsUser=`***operating_system_user* or **`instance.canSetPeOSCapabilities=true`** setting allows PEs to run as a specified user or with special operating system capabilities. These settings have implications for system security in a domain. In addition, a Streams application bundle file (.sab file) or toolkit operator model might contain capability specifications that allow the PEs to run with capabilities such as increased operating system user privileges. If you are using these settings:

- Only a root user can register a resource to run as a system service.
- Any Streams user who can create an instance in a domain can set the **`instance.runAsUser`** and **`instance.canSetPeOSCapabilities`** properties. By default, this is a user with DomainAdmistrator role.
- Any Streams user that can set instance properties can set the **`instance.runAsUser`** property. By default this is a user with the DomainAdmistrator or the InstanceAdministrator role.
- Any Streams user who can set domain properties can set the **`instance.canSetPeOSCapabilities`** property for an instance. By default, this is a user with the DomainAdmistrator role.
- Any Streams user that can submit a job to an instance with the **`instance.runAsUser`** and **`instance.canSetPeOSCapabilities`** properties set can have a job run on a resource with increased operating system user privileges. By default, this is a user with the DomainAdmistrator, InstanceAdministrator, or InstanceUser role.

- It is important to use sufficient file system permissions on a Streams application bundle (.sab) file that is submitted to an instance with the **instance.runAsUser** and **instance.canSetPeOSCapabilities** properties set. Ensure that the file system permissions restrict which users can provide or modify application bundle files.

9.4. Changing the cryptographic protocol for Streams services

Many domain and instance services support connections that use Transport Layer Security (TLS) cryptographic protocols. You can specify which cryptographic protocols the services use for secure communication by setting domain and instance properties. Starting with Streams Version 7.1.0 the default setting for Streams is TLSv1.2, which indicates that TLS 1.2 or later protocols are used.

About this task

You can specify the cryptographic protocol for the domain and instance services in Table 30 and Table 31.

Notes:

- The **domain.sslOption** domain property is used as the default value for the **sslOption** properties that are listed.
- If you set the **sws.sslProtocol** property to TLSv1.2, you must also specify one of the following settings:
  - Set the **domain.sslOption** property to TLSv1.2.
  - Set both the **aas.sslOption** and **jmx.sslOption** properties to TLSv1.2.
- If the **domain.sslOption** property is set to TLSv1.2, the WebSphere® Application Server **com.ibm.jsse2.sp800-131** system property is set to strict. For more information about SP800-131 standard strict mode, see Configuring WebSphere Application Server for SP800-131 standard strict mode.

Table 9.8. Domain services

Service name	Domain property name
authentication and authorization service	**aas.sslOption**
domain controller service	**controller.sslOption**
web management service	**sws.sslProtocol**

Table 9.9. Instance services

Service name	Instance property name
application deployment service	**app.sslOption**
application manager service	**sam.sslOption**
application metrics service	**srm.sslOption**

Service name	Instance property name
view service	**`view.sslOption`**

The domain and instance properties can have the following values:

- TLSv1.2 indicates that the service uses only TLS 1.2 or later protocols. If a TLS 1.2 connection cannot be established, it does not fall back to lower versions of TLS support. Starting with Streams Version 7.1.0 TLSv1.2 is the default value.
- none indicates that the service does not use TLS or SSL. You cannot specify this value for the **`sws.sslProtocol`** domain property.

Note

TLS 1.2 or later is recommended for Streams. This version is more secure than earlier TLS versions and SSL.

The **`sws.sslProtocol`** domain property has an extra value: useJavaSetting. This property indicates that the web management service supports the cryptographic protocols that are specified by the Java configuration of processes that connect to the service. This value is the default value.

For more information about these properties, run **`streamtool man domainproperties`** and **`streamtool man properties`**.

Tip

Before you change the cryptographic protocol, consider which Streams interfaces you use and how they are affected. For example, you must open the Streams Console in a web browser that supports the same cryptographic protocols that you specify for the web management service.

Procedure

You can specify the cryptographic protocol when you create or update a domain or instance. In Streams Studio, you can specify the cryptographic protocol when you add or edit a domain connection.

- Specify the cryptographic protocol when you create the domain or instance.
  - Use the Streams Console or the **`streamtool mkinstance`** command to create an instance and specify the appropriate instance property.
- Specify the cryptographic protocol after you create the domain or instance.
  - To update a domain property, use the **`streamtool setdomainproperty`** command, then restart the domain.
  - To update an instance property, use the Streams Console or the **`streamtool setproperty`** command, then restart the instance.
- In Streams Studio, specify the cryptographic protocol when you add or edit a domain connection. The protocol that you specify must match the value on the **`jmx.sslProtocol`** domain property. To obtain that value, run the **`streamtool getdomainproperty jmx.sslOption`** command.

- When you add a domain connection, use the *TLS Protocol* list to select the cryptography protocol that matches the domain configuration.
- If you have a domain connection defined for the domain, use the Edit Domain Connection window to update the cryptography protocol. For the change to take effect, disconnect and then reconnect to the domain.

Related tasks:

"Setting up client certificate authentication for Streams users" Use this procedure to set up client certificate authentication for a Streams domain by using X.509 certificates. Using client certificate authentication is optional.

Adding domain connections for Streams Studio

To use Streams Studio with Streams, you must add at least one domain connection.

9.5. Securing a Streams cluster

While applications are deployed into production environments with specific recommendations that apply to individual customers, enterprise organizations generally have security policies and procedures that must be met and followed. To meet enterprise requirements, a combination of product and environment configuration is needed. The following are some key features using the standard information security CIA triad.

Maintaining confidentiality:

- **Granular access control:** Streams uses access control lists (ACLs) to manage user authorization for domains and instances. An ACL contains the type of domain and instance objects to secure and the actions that a user or group is authorized to perform against the object.
- **Encryption:** All connections to the Streams Console and REST APIs use the HTTPS protocol. Furthermore, you can encrypt PE to PE communication and management communication with TLS1.2. Most toolkits have operators that enable the sourcing and sinking of data using TLS1.2.
- **Authentication:** The default user authentication method is PAM or LDAP. For a basic domain, Streams uses PAM. For an enterprise domain, you can specify either LDAP or PAM as the default method when you create the domain, and then customize user authentication after the domain is created. You can also create JAAS plug-in modules and integrate with any back-end authentication system.

Ensuring integrity:

- **Checkpointing:** Operator state can be persisted at run-time to allow recovery from a failure. This enables you to design applications to guarantee delivery of data.
- **Auditability:** Streams supports comprehensive, multiple-level auditing of product and user operations.

Sustaining availability:

- **Failover:** There is a high availability count property in both domains and instances. This property indicates how many copies of management services are maintained by a domain or instance. If a leader service fails, the product automatically fails over to another instantiation of the service.

- **Redundancy:** The Streams product tightly integrates with redundant LDAP servers, Zookeeper quorums, and multiple Redis servers.

In addition to the security features that are included in the product, the Streams product is developed and maintained following IBM Secure Engineering practices. This includes IBM's Product Security Incident Response Team (PSIRT) process. You can review this process at https://www.ibm.com/security/secure-engineering/process.html. The Streams code base is scanned using the AppScan product suite (https://www.ibm.com/security/application-security/appscan). All high or medium issues that are not resolved before the product is made publicly available are opened as PSIRT records and resolved following the IBM PSIRT process.

Related concepts:

"User authorization for Streams"

Streams uses access control lists (ACLs) to manage user authorization for domains and instances. An ACL contains the type of domain and instance objects to secure and the actions that a user or group is authorized to perform against the object. The ACLs are initialized when you create a domain or instance. "User authentication options for Streams"

The default user authentication method for Streams domains is PAM or LDAP. For a basic domain, Streams uses PAM. For an enterprise domain, you can specify either LDAP or PAM as the default method when you create the domain, and then customize user authentication after the domain is created.

Chapter 10, "Configuring audit logging for Streams,"

Streams supports comprehensive, multiple-level auditing of product and user operations. By default, audit logging is not enabled.

Related tasks:

Chapter 9, "Configuring security for Streams,"

Each Streams domain and instance maintains its own security configuration.

"Changing the cryptographic protocol for Streams services" Many domain and instance services support connections that use Transport Layer Security (TLS) cryptographic protocols. You can specify which cryptographic protocols the services use for secure communication by setting domain and instance properties. Starting with Streams Version 7.1.0 the default setting for Streams is TLSv1.2, which indicates that TLS 1.2 or later protocols are used.

Chapter 13, "Configuring a checkpoint data store for Streams domains and instances,"

Use this procedure to configure a checkpoint data store for Streams domains and instances. A checkpoint data store is used by applications that call the Streams Checkpointing API.

Chapter 8, "Setting up a Streams enterprise domain on multiple resources,"

An *enterprise domain* can have multiple resources and users. This type of domain is typically used for production environments. You can configure high availability to ensure that Streams can continue to run even if resources fail or are not available.

Chapter 10. Configuring audit logging for Streams

Streams supports comprehensive, multiple-level auditing of product and user operations. By default, audit logging is not enabled.

When audit logging is enabled, the following types of events are logged:

- All user operations from the Streams **`streamtool`** command-line interface and the JMX API.
- All permission checking performed by Streams.
- Any event that changes the state of a Streams domain, instance, or resource.

10.1. Enabling audit logging for Streams

To enable audit logging for Streams, use this procedure to update the required domain properties and create a log4j configuration file.

About this task

If audit logging is enabled by using the default file appender, the default log file name is streams.audit.log. This log file is located in the same directory as all other Streams log and trace files, which is the directory that is specified on the **`domainLog.path`** property.

For more information about the audit logging properties, enter streamtool man domainproperties. All properties that apply to audit logging are prefixed with **`auditlog`**. To set properties, use the **`streamtool setdomainproperties`** command.

Procedure

1. Set the **`auditlog.level`** domain property to standard. The default setting is off.
2. Create a `log4j` configuration file for audit logging. The following sample file is installed with Streams:

   `product-installation-root-directory``/7.1.0.0/etc/trc/log.default.properties.xml`

   Important

   Do not edit the sample `.xml` file. Make a copy of the sample file and then edit the copy as needed. If you edit the sample `.xml` file and it becomes corrupted, errors can occur.

   ---

3. Optional: Customize audit logging by adding filters to your `log4j` configuration file.
4. Optional: Change the default location of the audit log files in your `log4j` configuration file.

   By default, Streams saves the audit log files in the same location as the other product log files. To save the audit log files in a different location, change the **`streams_log_path`** variable in your `log4j` configuration file to another path on your system.

5. Set the **auditlog.log4jPropertiesFile** domain property to point to your log4j configuration file. The log4j configuration file must be accessible in the path that is specified by the **auditlog.log4jPropertiesFile** property by all resources that run the audit log service. This can be a shared file location or an exact copy on each resource.

6. Optional: Set the **auditlog.log4jAppenderClassPath** domain property. This property is used to find the customized log4j appender ClassPath.

7. If the domain is running, you must restart the domain for the changes to be effective.

Results

After the changes become effective for the domain, users can dynamically enable or disable audit logging by updating the **auditlog.level** domain property. Possible settings are off (no auditing) or standard (auditing performed at the standard level).

10.2. Customizing audit logging for Streams by adding filters

The sample log4j configuration file that is installed with Streams includes an appender example that shows you how to filter out audit event entries and messages based on certain conditions.

Before you begin

Enable audit logging and create a log4j configuration file.

Procedure

Optional: Add filters to your log4j configuration file. Examples of Streams filters are:

- domainID, instanceID, hostname, and userID:
- components such as streamtool, jmx, api, and sam
- operations such as submitjob, addhost, and getinstancestate
- combination of any of the above

The following example in the sample file includes audit entries for all log messages for **streamtool** and **submitjob** operations:

```
<appender name="auditAppender" class="org.apache.log4j.FileAppender">
    <param name="File" value="${streams_log_path}/logs/streams.audit.log"/>
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="%m%n"/>
        </layout>
    <filter class="org.apache.log4j.filter.ExpressionFilter">
        <param name="ConvertInFixToPostFix" value="true" />
        <param name="AcceptOnMatch" value="true" />
        <param name="Expression" value="MSG ⌂= 'SRC[streamtool]' || MSG ⌂=
 'OP[submitjob'" />
    </filter>
    <filter class="org.apache.log4j.filter.DenyAllFilter" />
</appender>
```

To help you create filter expressions, see the audit log examples.

10.3. Streams audit log examples

Audit log entries contain information about Streams domains, instances, users, operations, and operation objects. Detailed messages can provide additional information.

Audit log format:

```
Timestamp* D[domain] I[instance] H[host] U[user] SUCCESS/FAILURE/NA
SRC[operation-source] OBJ[operation-object] OP[operation-and-
parameters] message
```

`Timestamp*` is the time that the Streams component performs operations and auditing. It is not the time that the logging service and log4j write the message to the appender. This format is the same as the Streams logging timestamp format, which is `yyyy-MM-dd'T'HH:mm:ss.SSSZ`.

The following examples are **`streamtool startinstance`** command operations. The first example contains a message that provides additional information.

```
2015-01-16T13:30:53.869-0500 D[StreamsDomain] I[StreamsInstance]
H[host1.ibm.com(192.0.2.0)] U[streamsadmin, streamsadmin] SUCCESS
SRC[api] OBJ[com.ibm.streams.admin.internal.api.StreamsInstance] OP[start,
authUser=streamsadmin@StreamsDomain
 listener=com.ibm.streams.cli.StartInstance@6b0cd689]
CDISA3004W The StreamsInstance instance started with warnings for the
 following domain:
StreamsDomain. The instance is operational.

2015-01-16T13:30:53.932-0500 D[StreamsDomain] I[StreamsInstance]
H[host2.ibm.com(198.51.100.0)] U[streamsadmin, streamsadmin] SUCCESS(0)
SRC[streamtool] OBJ[com.ibm.streams.cli.StartInstance] OP[startinstance,
 d=StreamsDomain
zkconnect=host3:2181,host4:2181,host5:2181 i=StreamsInstance
```

Chapter 11. Configuring transport mechanisms for Streams

Streams supports several high-performance and configurable transport mechanisms for communicating across processing elements (PEs). You can configure the transport mechanism by updating Streams properties or by specifying options in stream processing applications.

Before you begin

Restrictions:

- If you use the LLM transport, do not enable encrypted connections between processing elements. If both the LLM transport and encrypted connections are enabled, the processing elements can't connect.

11.1. Configuring transport mechanisms by using Streams properties

You can use domain and instance properties to configure transport mechanisms for Streams.

Before you begin

Restrictions:

- If you use the LLM transport, do not enable encrypted connections between processing elements. If both the LLM transport and encrypted connections are enabled, the processing elements can't connect.

11.1.1. Application network properties for Streams domains and instances

Use the **`domain.applicationNetwork`** and **`instance.applicationNetwork`** properties to specify the subnetworks that are used by the Streams data transport mechanism for instance jobs and PEs.

By default, Streams uses the IP address that is configured in the Domain Name System or in the /etc/hosts file.

To specify the **`domain.applicationNetwork`** or **`instance.applicationNetwork`** property value, use Classless Inter-Domain Routing (CIDR) notation, which includes the combination of both an IP address and a mask bit value in the following format:

`IP-address/mask-bit-value`

In the following example, the IP address is 192.0.2.0 and the mask bit value is 24, which converts to a subnet mask of 255.255.255.0:

```
instance.applicationNetwork=192.0.2.0/24
```

To specify multiple CIDR values, separate the values with a comma as show in the following example:

```
instance.applicationNetwork=192.0.2.0/24,198.51.100.0/24
```

For jobs and PEs, Streams checks the CIDR values in the order specified, and uses the first match that is found for the connection. If no match is found, Streams uses the IP address that is configured in the Domain Name System or in the /etc/hosts file.

For system services, Streams checks the CIDR values in the order specified, and attempts to select an alternate interface with a subnetwork that is not in the subnetworks that are specified by the application network properties.

- Domain services use the *domain.applicationNetwork* property.
- Instance services use the *instance.applicationNetwork* property.

If an alternate interface cannot be found, Streams uses the IP address that is configured in the Domain Name System or in the `/etc/hosts` file.

If you are using the LLM_RUM_IB transport, Streams checks both the **`instanceNetwork.llmInfinibandInterfaceName`** instance property and the **`instance.applicationNetwork`** instance property to determine the IP address that is used.

To display or set property values for domains and instances, use the following **`streamtool`** commands:

- For the **`domain.applicationNetwork`** property, use the **`streamtool getdomainproperty`** command to display the property value and the **`streamtool setdomainproperty`** command to update the value.
- For the **`instance.applicationNetwork`** property, use the **`streamtool getproperty`** command to display the property value and the **`streamtool setproperty`** command to update the value.

Related reference:

streamtool getproperty command streamtool setproperty command

11.1.2. Encrypted PE connections property for Streams instances

Use the **`instance.transportSecurityType`** property to enable or disable encrypted connections between processing elements (PEs). By default, connections between PEs are not encrypted.

Before setting the **`transportSecurityType`** property, review the performance considerations.

To enable encryption between PEs, specify one of the following supported Transport Layer Security (TLS) cryptographic protocols:

- TLSv1.2 for TLS 1.2 or later protocols

Restrictions:

- If you use the LLM transport, do not enable encrypted connections between processing elements. If both the LLM transport and encrypted connections are enabled, the processing elements can't connect.

Notes:

- TLS 1.2 or later is recommended for Streams. This version is more secure than earlier TLS versions and SSL.
- Because of a security vulnerability, the IBM SDK that is installed with Streams disables SSL 3.0. For more information, see the following IBM Support documentation: IBM SDK, Java Technology Edition fixes to mitigate against the POODLE security vulnerability (CVE-2014-3566).

To display the value of the **`instance.transportSecurityType`** property for an instance, use the **`streamtool getproperty`** command. To set this property for an instance, use the **`streamtool setproperty`** command.

The following example specifies that TLS 1.2 or later is used to encrypt connections between PEs:

```
streamtool setproperty instance.transportSecurityType=TLSv1.2
```

The following example shows how to disable encryption:

```
streamtool setproperty instance.transportSecurityType=none
```

Note

If Streams cannot establish a connection with the specified protocol, check the PE and processing element container service (PEC) logs. The following examples are typical causes of failure to connect errors:

- The openssl RPM is not installed. This RPM is required for an encrypted connection. Ensure that you install the version of the openssl RPM that is required by Streams and checked by the dependency checker script.

  In this case, the PEC crashes with an informational message. If you run the **`streamtool lspes`** command, the PE is in Stopped state.
- A network failure occurs.

  In this case, Streams continues to retry the connection until it succeeds.
- The target PE does not authenticate with the proper security protocol.

  In this case, Streams continues to retry the connection until it succeeds.

Related reference:

Streamtool commands

This reference section provides a description of each **`streamtool`** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

Log and trace files generated by Streams

Required RPMs for Streams

11.1.2.1. Performance considerations for encrypted PE connections

The SSL/TLS value that you specify for the **`transportSecurityType`** property can impact PE connection performance. Streams supports TLS v1.2 and later protocols.

Note

TLS 1.2 or later is recommended for Streams. This version is more secure than earlier TLS versions and SSL.

For encrypted PE connections, Streams uses OpenSSL, RSA 2K encryption, and the Advanced Encryption Standard (AES) cipher. These operations can consume a large amount of CPU time that might not be consumed if you use TCP connections.

Performance can vary significantly depending on your hardware. You can use the following **`openssl speed`** commands to obtain performance information for your specific hardware.

- For performance information during RSA 2K handshakes, run the following command:

  ```
  openssl speed rsa2048
  ```

  Example:

  ```
  Doing 2048 bit private rsa's for 10s: 4830 2048 bit private RSA's in
   10.00s
  Doing 2048 bit public rsa's for 10s: 151997 2048 bit public RSA's in
   10.00s
  OpenSSL 1.0.1e-fips 11 Feb 2013
  . . .
  sign verify sign/s verify/s
  rsa 2048 bits 0.002070s 0.000066s 483.0 15199.7
  ```

- For performance information during AES encryption, run the following command:

  ```
  openssl speed aes
  ```

  Example:

  ```
   Doing  aes-128  cbc  for  3s  on  16 size blocks: 16021714 aes-128 cbc's in 3.00s
   Doing  aes-128  cbc  for  3s  on  64 size blocks: 4300801 aes-128 cbc's in 3.00s
   Doing  aes-128  cbc  for  3s  on  256 size blocks: 1097539 aes-128 cbc's in 2.99s
   Doing  aes-128  cbc  for  3s  on  1024 size blocks: 275925 aes-128 cbc's in 3.00s
   Doing  aes-128  cbc  for  3s  on  8192 size blocks: 34471 aes-128 cbc's in 3.00s
   Doing  aes-192  cbc  for  3s  on  16 size blocks: 13556034 aes-192 cbc's in 3.00s
   Doing  aes-192  cbc  for  3s  on  64 size blocks: 3598405 aes-192 cbc's in 3.00s
   Doing  aes-192  cbc  for  3s  on  256 size blocks: 913409 aes-192 cbc's in 3.00s
   Doing  aes-192  cbc  for  3s  on  1024 size blocks: 229628 aes-192 cbc's in 3.00s
   Doing  aes-192  cbc  for  3s  on  8192 size blocks: 28710 aes-192 cbc's in 3.00s
   Doing  aes-256  cbc  for  3s  on  16 size blocks: 11703042 aes-256 cbc's in 3.00s
   Doing  aes-256  cbc  for  3s  on  64 size blocks: 3092630 aes-256 cbc's in 3.00s
   Doing  aes-256  cbc  for  3s  on  256 size blocks: 783286 aes-256 cbc's in 3.00s
   Doing  aes-256  cbc  for  3s  on  1024 size blocks: 196494 aes-256 cbc's in 2.99s
   Doing  aes-256  cbc  for  3s  on  8192 size blocks: 24566 aes-256 cbc's in 3.00s
  OpenSSL 1.0.1e-fips 11 Feb 2013The 'numbers' are in 1000s of bytes per second processed.
  type 16 bytes 64 bytes 256 bytes 1024 bytes 8192 bytes
  aes-128  cbc  85449.14k  91750.42k  93969.89k  94182.40k  94128.81k
  aes-192  cbc  72298.85k  76765.97k  77944.23k  78379.69k  78397.44k
  aes-256  cbc  62416.22k  65976.11k  66840.41k  67294.27k  67081.56k
  ```

. . .

If your applications are experiencing reduced throughput with encrypted PE connections, the following suggestions might improve application performance:

- Use the Streams fusion option for applications to reduce the number of PE to PE connections. Fusing PEs can reduce the number of encryption operations needed and allow more data to flow through the PE.
- Change the application so that it sends fewer or larger tuples. With SSL/TLS, there are fixed costs associated with every tuple that is sent. Sending fewer or larger tuples can reduce the processing time for this overhead.
- Use a hardware accelerator such as an x86 processor with the Intel Advanced Encryption Standard New Instructions (AES-NI) feature. Streams can also use hardware accelerators with the OpenSSL update that adds another decryption engine.

Note

Not all versions of OpenSSL support AES-NI. If you install the version of the openssl RPM that is required by Streams and checked by the dependency checker script, Streams supports and uses AES-NI.

Related tasks:

Specifying how operators are fused when you submit a job

Related reference:

Required RPMs for Streams

Related information:

OpenSSL documentation

11.2. Configuring transport mechanisms for Streams in stream processing applications

You can specify a global configuration for an application or specify different transport mechanisms for individual operators in the application.

About this task

The **sc** compiler provides the **-F, --use-transport** option for setting the transport mechanism to use globally for an application. Individual SPL operators can be configured to override the global configuration selected by the **-F** option and use different transports for their inputs and outputs.

Procedure

- To configure the TCP transport, use the **-F tcp** option. This option is the default if the **-F** option is not explicitly used. To enable secure TCP connections, set the **instance.transportSecurityType** property to the desired security type with the **streamtool** command.

- To configure the IBM WebSphere MQ LLM Reliable Unicast Messaging transport with TCP, use the **-F llm_rum_tcp** option. This is a high-performance, low-latency transport, which can be configured.
- To configure the IBM WebSphere MQ LLM Reliable Unicast Messaging transport with InfiniBand, use the **-F llm_rum_ib** option. This is a high-performance low-latency transport, which can be configured.

Related reference:

sc command

This command is used to compile streams processing applications.

spl-transport-stats command

This command creates a transport statistics model by running a transport performance test.

Chapter 12. Configuring Streams Studio for remote development

You can install Streams Studio on a local Linux or Windows system that acts as a client and install Streams on a remote Linux system that acts as a server. To connect to the remote Linux system where Streams is installed, you must configure a connection.

Before you begin

Install Streams Studio on your local Linux or Windows system. For instructions, see Installing Streams Studio for remote development.

Ensure that you have a user account on the remote Linux system where Streams is installed.

Streams Studio uses Remote System Explorer (RSE) to provide connectivity to the remote Linux system for building stream processing applications. When you run Streams Studio for the first time on the local Linux or Windows system, you must configure a connection to specify how you want to start the RSE server on the remote system and establish a connection. You have two options for configuring a connection:

- Create a connection configuration file by editing the `default.studioconfig` sample file in your Streams Studio installation directory. When you start Streams Studio in the following procedure, the configuration values in this file are automatically entered for you.
- Specify the configuration values when you complete the following procedure.

If Streams Studio has problems accessing the Streams domain, see the information for clients that are external to the Streams cluster in Firewall configuration guidelines for Streams.

Procedure

1. Start Streams Studio on your local system.
   - On a Linux system, go to the directory where you installed Streams Studio and enter `./streamsStudio`.
   - On a Windows system, go to the directory where you installed Streams Studio and double-click streamsStudio.exe.
2. In the Workspace Launcher window, specify the local workspace where you want to store your SPL projects, and then click **OK**. This workspace is a directory on the local system where you installed Streams Studio.
3. In the Streams Install Location Details window:
   - If you are using a connection configuration file, the connection to the remote Linux system is automatically created and displayed in the **Host connection** field. This connection is based on the values in your `.studioconfig` file.
   - If you are not using a connection configuration file, click *New* to create a connection to the remote Linux system.
     a. In the Select Remote System Type window, select the remote Linux system type, and then click **Next**.

b. In the New Connection window, specify the following values, and then click **Next**.

   - Parent profile name. RSE creates a default profile that typically uses the name of the local system. Accept the default profile or specify an RSE profile of your choice.
   - Host name or IP address of the remote Linux system where Streams is installed
   - Connection name that is unique to your profile. This name will be displayed in the tree view in the Remote Systems view.
   - Optional description
   - Optional proxy settings for the remote Linux system

c. In the New remote connection configuration window, configure the connection properties to define how you want to connect to the RSE server, and then click **Finish**.

   - **Remote daemon** option: RSE establishes the connection by using the remote daemon. The remote daemon must be running on the remote Linux system at a predefined port and must be started by a root user.

     Enter the following commands to start the remote daemon:

     ```
     su -l root
     cd product-installation-root-directory/7.1.0.0/etc/rseserver
     perl ./daemon.pl [daemonPort] [serverPortRange]
     ```

     where *`product-installation-root-directory`* is the root directory for the Streams product installation.

     ## Notes:

     - By default, the server daemon runs on port 8050. To use a different port, you can specify the optional daemonPort argument. If your daemon runs behind a firewall, you might want to use the optional serverPortRange argument to restrict selected server ports to the range that you specify. For example, **`perl ./daemon.pl 4075 10000-10010`** specifies a port range of 10000-10010.
     - The server daemon uses the Perl script `auth.pl` to authenticate the user who is making the connection. The `auth.pl` script uses basic password authentication by using the password file. You must update the script to use the authentication mechanism that matches your system.

   - **SSH** option: RSE establishes the connection by using Secure Shell (SSH) support. To establish a connection, the SSH command, **`./server.sh`**, is issued to start the RSE server. You must specify the path where the RSE server is installed on the remote Linux system.

     ## Note

     If your remote Linux system is behind a firewall, you might want to restrict the server to use a specified port range by setting the port variable in the server.sh file. To set

the port variable, edit the server.sh file on your remote Linux system and set the port variable to the desired port range. For example, port=10000-10010; specifies a port range of 10000-10010.

In the *Path to installed server on host* field, specify an absolute path or a path relative to your home directory, as shown in the following examples:

```
/home/user-directory/product-installation-root-directory/7.1.0.0/etc/rseserver
product-installation-root-directory/7.1.0.0/etc/rseserver
```

4. When prompted, enter your user ID and password.
5. To specify the location where Streams is installed, click **Browse** and select the directory where Streams is installed on the remote Linux system.

   Note

   If this information is specified in the connection configuration file, the value is already entered.

6. To specify the remote workspace, click **Browse**, expand **My Home**, and then select a folder on the remote Linux system. To create a folder on the remote Linux system from your local system, select the parent folder, and click **New** > **Folder**.

Results

The Remote Systems view displays the short name of the connection with the following nodes under the connection:

- **Files**
- **Processes**
- **Contexts**
- **Shells**
- **Ssh Terminals**

What to do next

After establishing the connection, you can access the file system and run remote commands on the remote Linux system. You can import, create, build, run, and monitor streams processing applications; create Streams runtime instances; and view instance and application graphs.

Related concepts:

Local and remote workspaces in Streams Studio

A local workspace is a directory that contains your Eclipse workspace. A remote workspace is a directory on the Linux system where Streams is installed. The remote workspace contains a copy of the SPL projects and artifacts that are in the local workspace.

12.1. Creating a connection configuration file for remote development

You can use the Streams Studio sample connection configuration file to create one or more configuration files for remote development.

Procedure

Edit the sample connection configuration file, `default.studioconfig`, that is shown in the following example. This file is in your Streams Studio installation directory.

```
# The connection configuration file must be in the Streams Studio
 installation directory.
# The connection configuration file must have a 'studioconfig' file
 extension for it to be recognized
# Multiple connection configuration files can be present.
# For each connection configuration file, a remote connection is
 automatically created at startup if one does not exist.
# The default connection used at initial startup is configured by
 'default.studioconfig'
#
# Remove the '#' at the beginning of the line to set the option.

# Remote Linux host name or IP address
#host=hostname.domain.com

# Absolute path for RSE server
# The RSE server is typically in <streams_install_location>/<version>/etc/
rseserver on the Linux host
#rseserver=/opt/ibm/InfoSphere_Streams/7.1.0.0/etc/rseserver

# (Optional) Use SSH tunnel option. If not specified, defaults to false.
#useSSHTunnel=true

# (Optional) Type of the remote system. Defaults to x86 Linux if not
 specified.
# Specify one of the following:
# - org.eclipse.rse.systemtype.x86linux (x86 Linux).
# - org.eclipse.rse.systemtype.powerlinux (Power Linux)
#systemtype=org.eclipse.rse.systemtype.x86linux
```

Chapter 13. Configuring a checkpoint data store for Streams domains and instances

Use this procedure to configure a checkpoint data store for Streams domains and instances. A checkpoint data store is used by applications that call the Streams Checkpointing API.

By default, no checkpoint data store is configured for domains and instances. You can use the data store that is included with Streams or another supported data store, such as Redis and IBM Cloud Object Storage.

You can also specify a Redis password to ensure a secure configuration.

About this task

Checkpointing is the process of persisting operator state at run time to allow recovery from a failure. If a failure occurs, the operator can be restarted by resetting from the checkpointed state. The operator uses the Checkpointing API to serialize its state data. The resulting checkpoint is stored in the checkpoint data store that you configure.

For each domain or instance, you choose the type and location of the data store that the Checkpointing API writes to. Any changes to the configuration do not take effect until the domain or instance is stopped and restarted. The domain property setting is automatically inherited by all the instances in that domain unless the instance property is set.

Notes:

- Any instance that runs applications that use checkpointing or consistent region features is required to specify a checkpoint data store by setting the **`instance.checkpointRepository`** and **`instance.checkpointRepositoryConfiguration`** properties. Typically, operators in a consistent region require checkpointing.
- A checkpoint data store is also required for applications that have operators with the config checkpoint clause.

The following properties are used to configure a checkpoint data store:

domain.checkpointRepository

If the **`instance.checkpointRepository`** property is not configured for an instance, this domain property specifies the default repository that Streams uses to store application checkpoint data for the instance.

This property can have the following values:

notSpecified

Default value. Indicates that no checkpoint data store is provided for the domain.

fileSystem

Indicates that the checkpoint data for the domain and its instances is stored in a file system directory.

redis

Indicates that the checkpoint data for the domain and its instances is stored in a Redis checkpoint data store.

ibmCloudObjectStorage

Indicates that the checkpoint data for the domain and its instances is stored in an IBM Cloud Object Storage checkpoint data store.

Before you run applications that call the Checkpointing API, you must configure your domain or instance to specify either `fileSystem`, `redis`, or `ibmCloudObjectStorage`.

Note

If the **`domain.checkpointRepository`** and **`instance.checkpointRepository`** properties are both set to `notSpecified`, the instances in the domain do not allow submitting a job that requires checkpointing.

---

domain.checkpointRepositoryConfiguration

If configuration information is not specified for an instance, this domain property specifies the default configuration information for the repository that Streams uses to store application checkpoint data for the instance. If the **`domain.checkpointRepository`** property is set to the default value of `notSpecified`, the **`domain.checkpointRepositoryConfiguration`** property is ignored.

This property contains a string with the following configuration information:

- For `fileSystem`, the string contains the absolute path to an existing directory, which is the root checkpointing directory.
- For `redis`, the string contains a list of `redis` server host names and port numbers.
- For `ibmCloudObjectStorage`, the string contains a JSON payload with Cloud Object Storage (COS) service credentials that include endpoint, bucket name, and configuration entry name that contains keys.

instance.checkpointRepository

This instance property specifies the repository that Streams uses to store application checkpoint information.

This property can have the following values:

notSpecified

Default value. Indicates that no checkpoint data store is provided for the instance.

fileSystem

Indicates that the checkpoint data for the instance is stored in a file system directory.

redis

Indicates that the checkpoint data for the instance is stored in a Redis checkpoint data store.

ibmCloudObjectStorage

Indicates that the checkpoint data for the instance is stored in a IBM Cloud Object Storage checkpoint data store.

Before you run applications that call the Checkpointing API, you must configure your domain or instance to specify either `fileSystem`, `redis`, or `ibmCloudObjectStorage`.

Notes:

- If you do not set the **`instance.checkpointRepository`** property, the value of the **`domain.checkpointRepository`** property is used for the instance.
- If the **`instance.checkpointRepository`** and **`domain.checkpointRepository`** properties are both set to `notSpecified`, the instance does not allow submitting a job that requires checkpointing.

---

instance.checkpointRepositoryConfiguration

This instance property specifies the configuration information for the repository that Streams uses to store application checkpoint data for the instance. If the **`instance.checkpointRepository`** property is set to the default value of `notSpecified`, the **`instance.checkpointRepositoryConfiguration`** property is ignored.

This property contains a string with the following configuration information:

- For `fileSystem`, the string contains the absolute path to an existing directory, which is the root checkpointing directory.
- For `redis`, the string contains a list of `redis` server host names and port numbers.
- For `ibmCloudObjectStorage`, the string contains a JSON payload with Cloud Object Storage (COS) service credentials that include endpoint, bucket name, and configuration entry name that contains keys.

For more information about domain and instance properties, use the **`streamtool man domainproperties`** and **`streamtool man properties`** commands.

Procedure

To configure a checkpoint data store, specify one of the following repository values:

- `fileSystem`

This value specifies that Streams saves checkpoint data in a file system directory.

For `fileSystem`, the **`domain.checkpointRepositoryConfiguration`** or **`instance.checkpointRepositoryConfiguration`** properties specify an absolute path to an existing directory, which is the root checkpointing directory. If the instance needs to run jobs with restartable and relocatable operators, then place the checkpointing directory in a shared file system.

The value of **`checkpointRepositoryConfiguration`** is in a JSON format.

```
instance.checkpointRepositoryConfiguration = " { \"Dir\" : path } "
```

`path` is the absolute path to an existing directory.

The following example sets the checkpoint store in the user's home directory.

```
instance.checkpointRepositoryConfiguration=" { \"Dir\" : \"/home/user/
ckpt\" } "
```

You can set domain and instance properties by using the **`streamtool setdomainproperty`** and **`streamtool setproperty`** commands. Use the **`streamtool getdomainproperty`** and **`streamtool getproperty`** commands to display property values.

- `redis`

  This value specifies that Streams uses a Redis checkpoint data store.

  Redis is an open source key-value data store. To use this store for checkpointing, you need to set up your Redis servers and provide their location to Streams with host names and ports in the **`checkpointRepositoryConfiguration`** string. Multiple Redis servers can be configured and checkpoint data is distributed among the servers through sharding and replication.

  The value of **`checkpointRepositoryConfiguration`** is in a JSON format.

  For `redis`, operators' checkpoint data are stored on one or multiple Redis servers through sharding and replication. The Redis servers are organized as K shards and N replica groups (K x N servers in total). Checkpoint data is distributed among the K shards, and each checkpoint data is replicated on N Redis servers. The number of shards (K) determines the capacity of the backend store. The number of replicas (N) determines the reliability of backend store. Provisioning 2n+1 replica groups can tolerate n server failures. The following example contains the JSON format for specifying the information of Redis servers.

  ```
  {
      "replicas" : N,
      "shards" : K,
      "replicaGroups" : [replicaGroup1, replicaGroup2, ..., replicaGroupN]
  }
  ```

  The value of "shards" (K) specifies number of shards and it must be a positive integer value. The value of "replicas" (N) specifies number of replica groups and it must be an odd integer value (for example, 3). The "`replicaGroups`" is an array of N JSON objects. Each JSON object in the `replicaGroups` array corresponds to a replica group and is specified as shown in the following example:

  ```
  {
  ```

```
    "servers" : [HostAndPortOfServer1, HostAndPortOfServer2, ...,
 HostAndPortOfServerK],
    "description" : StringOfOptionalDescription
}
```

The value of "`servers`" is a JSON array that contains K "`host:port`" for the K Redis servers in the replica group. `host` can be host name or IP address and port is the port number that is used by the Redis server. The "`description`" is optional.

As an example, the setup for 24 Redis servers with 8 shards and 3 replicas has the following configuration value.

```
instance.checkpointRepositoryConfiguration=" {
\"replicas\" : 3,
\"shards\" : 8,
\"replicaGroups\" : [
    { \"servers\" : [\"host0:port0\", \"host1:port1\", \"host2:port2\",
 \"host3:port3\",
      \"host4:port4\", \"host5:port5\",\"host6:port6\", \"host7:port7\"],
      \"description\" : \"rack1\" },
    { \"servers\" : [\"host8:port8\", \"host9:port9\", \"host10:port10\",
 \"host11:port11\",
      \"host12:port12\",\"host13:port13\", \"host14:port14\",
 \"host15:port15\"],
      \"description\" : \"rack2\" },
    { \"servers\" : [\"host16:port16\", \"host17:port17\", \"host18:port18\",
      \"host19:port19\", \"host20:port20\",\"host21:port21\",
 \"host22:port22\",
      \"host23:port23\"], \"description\" : \"rack3\" }
]
}"
```

- `redis` for a secure platform configuration

  You can configure Streams to use Redis servers which have authentication enabled as a checkpoint data store. You can specify a Redis password as part of the domain or instance **`checkpointRepositoryConfiguration`** property, store this password, and pass it to SPL runtime in a secure way:

  1. Use the **`streamtool setproperty`** command to set the **`checkpointRepository`** string to Redis:

     ```
     streamtool setproperty -d $STREAMS_DOMAIN_ID -i $STREAMS_INSTANCE_ID checkpointRepository=redis
     ```

  2. Use the **`streamtool setrestrictedconfig`** command to store Redis passwords. If there are six Redis servers, for example, and every two of them share the same password, you can choose which property name you want to use to store the passwords, such as:

     ```
     streamtool setrestrictedconfig -d $STREAMS_DOMAIN_ID -i $STREAMS_INSTANCE_ID
      redisPassword1=passwdOne

     streamtool setrestrictedconfig -d $STREAMS_DOMAIN_ID -i $STREAMS_INSTANCE_ID
      redisPassword2=passwdTwo

     streamtool setrestrictedconfig -d $STREAMS_DOMAIN_ID -i $STREAMS_INSTANCE_ID
      redisPassword3=passwdThree
     ```

  3. Use the **`streamtool setproperty`** command to specify the normal **`checkpointRepositoryConfiguration`** property with the host, port, and password of the Redis servers:

     ```
     streamtool setproperty -d $STREAMS_DOMAIN_ID -i $STREAMS_INSTANCE_ID
      checkpointRepositoryConfiguration =" {
     ```

```
\"shards\":2,
\"replicas\":3,\
\"replicaGroups\":[\
  {\"servers\":[\"host1:6379:redisPassword1\",\"host2:6379:redisPassword1\",
 \"host3:6379:redisPassword2\"]},\
  {\"servers\":[\"host4:6379:redisPassword2\",\"host5:6379:redisPassword3\",
 \"host6:6379:redisPassword3\"]} \
  ] \
} "
```

- `ibmCloudObjectStorage`

  This value specifies that Streams uses a IBM Cloud Object Storage checkpoint data store.

  Cloud Object Storage is a highly scalable cloud storage service, designed for high durability, resiliency and security. To use this data store for checkpointing, you must set up your Cloud Object Storage service in IBM Cloud. For more information on how to create a Cloud Object Storage service, see the Cloud Object Storage product documentation .

  After you create the service, you must provide service credentials to Streams. These service credentials include endpoint, bucket name, and configuration entry name that contains service keys, as part of the string for **`checkpointRepositoryConfiguration`** in JSON format. You can find the service credentials on the Cloud Object Storage service dashboard in IBM Cloud.IBM Cloud Object Storage.

  For `ibmCloudObjectStorage`, operators' checkpoint data are stored on one or multiple Cloud Object Storage service buckets. Each operator puts objects with keys named in the form of *domainID-instanceID-jobID-operatorGlobalIndex* under a single bucket in the configured endpoint. The following example contains the JSON format for specifying the information of the Cloud Object Storage endpoint, bucket, and configuration entry containing the service keys:

  ```
  {
      "endpoint" : url,
      "bucket" : name,
      "auth" : name
  }
  ```

  The value of "`endpoint`" is a string that specifies the URL of a private or public Cloud Object Storage S3 endpoint. The value of "`bucket`" is a string that specifies the name of the container to store checkpoint objects. The bucket parameter is optional. If not specified, streams-checkpoint is used as bucket name. If the bucket does not exist, a new bucket named `streams-checkpoint` is created. The value of "`auth`" is a string that specifies the configuration entry name that contains all key-related information. Objects in Cloud Object Storage are encrypted at rest. This technology individually encrypts each object using per-object generated keys.

  The following example contains sample values for the JSON format you need to specify to use ibmCloudObjectStorage as the checkpoint data store.

  ```
  {
      "endpoint" : "s3-api.us-geo.objectstorage.service.networklayer.com",
      "bucket" : "streams-checkpoint",
      "auth" : "apiKey"
  }
  ```

Related reference:

Checkpointing

Checkpointing is the process of persisting operator state at run time to allow recovery from a failure. In case of failure, the operator can be restarted by resetting from the checkpointed state.

Streamtool commands

This reference section provides a description of each **`streamtool`** command, which includes details about command syntax and options. To obtain the help information for a specific command, enter `streamtool man` *`command-name`*.

Notices

This information was developed for products and services offered in the US.

21st Century Software Technologies, Inc. (“21CS”) may not offer the products, services, or features discussed in this document in other countries. Consult your 21CS representative for information on the products and services currently available in your area. Any reference to an 21CS product, program, or service is not intended to state or imply that only that 21CS product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any 21CS or IBM intellectual property rights may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-21CS product, program, or service.

21CS may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

21CS
6 Liberty Square, PMB 537
Boston, MA 02109-5800
US

21ST CENTURY SOFTWARE TECHNOLOGIES PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. 21CS may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-21CS websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this 21CS product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

21CS
6 Liberty Square, PMB 537
Boston, MA 02109-5800
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by 21CS under terms of the 21CS Software License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-21CS products was obtained from the suppliers of those products, their published announcements or other publicly available sources. 21CS has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-21CS products.

Questions on the capabilities of non-21CS products should be addressed to the suppliers of those products.

Statements regarding 21CS's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to 21CS or, IBM for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. 21CS and IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. Neither 21CS nor IBM shall be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year).

Portions of this code are derived from 21st Century Software Technologies, Inc. and IBM Corp. Sample Programs.

© Copyright 21st Century Software Technologies, Inc. © Copyright IBM Corp. _enter the year or years_.