EF Portal Administrator Guide

We'd Like to Hear from You

You can help us make this document better by telling us what you think of the content, organization, and usefulness of the information. If you find an error or just want to make a suggestion for improving this document, address your comments to <support@ni-sp-software.com>.

For product support, contact <support@ni-sp-software.com>.

Although the information in this document has been carefully reviewed, NI SP Software GmbH doesn't warrant it to be free of errors or omissions. NI SP reserves the right to make corrections, updates, revisions, or changes to the information in this document.

UNLESS OTHERWISE EXPRESSLY STATED BY NI SP, THE PROGRAM DESCRIBED IN THIS DOCUMENT IS PROVIDED "AS IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL NI SP BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING WITHOUT LIMITATION ANY LOST PROFITS, DATA, OR SAVINGS, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS PROGRAM.

Document Redistribution and Translation

Trademarks

America and other jurisdictions.

HP® is a registered trademark of HP Inc.

Google™ and Chrome™ are trademarks of Google Inc.

Red Hat® is a trademark of Red Hat, Inc.

SUSE® is a registered trademark of SUSE Linux AG.

Other names mentioned in this document may be trademarks of their respective owners.

Latest Version can be found at : https://www.ni-sp-software.com

Welcome

About this guide

This guide describes how you can install, configure, and manage a NI SP EF Portal instance.

Who this guide is for

This guide is intended for system administrators that install and manage one or more NI SP EF Portal instances.

What you should know

This guide assumes the following:

- You're familiar with Unix system administration tasks such as creating user accounts, sharing and mounting Network File System (NFS) partitions, and backing up the system.
- You have a foundation in web-related technologies such as the HTTP protocol, the SSL protocol, and the XML language.

Learn about NI SP Software products

World Wide Web

The website https://www.ni-sp-software.com has the latest information about NI SP EF Portal.

The same website also has information about other NI SP products and about the professional services provided by NI SP.

Get technical support

Contact NI SP or your EF Portal reseller for technical support.
Leverage the NI SP Knowledge Base AI assistant: https://www.ni-sp.com/ni-sp-support-kb-qa/

NI SP support contacts

Use one of the following to contact NI SP technical support.

Email : <support@ni-sp-software.com>
Web: https://www.ni-sp-software.com

When contacting the NI SP support team, include the full name of your company.

Collect support information

Use the "*Support/Collect support info*" service that's described in Operational Dashboard to collect some preliminary data to help NI SP support process your support request.

The output of this service is a compressed archive that contains all the gathered information. Send the compressed archive to the NI SP support team attached to your request.

Getting started

Topics

About NI SP EF Portal

EF Portal is a grid-enabled application portal for user-friendly HPC job submission, control, and monitoring. It includes sophisticated data management for all stages of job lifetime. It's integrated with most important job schedulers and middleware tools to submit, monitor, and manage jobs.

EF Portal provides a modular system where you can easily add new functionality, such as application integrations, authentication sources, and license monitoring. It also features a sophisticated REST API that you can use to enhance existing applications and develop custom solutions for your own environment.

EF Portal is a computing portal that uses existing scripting solutions when available. This means that, while using EF Portal, you can avoid interacting with a command line interface in situations where you might have not had that option before.

Based on the latest and most advanced Web 2.0 standards, it provides a flexible infrastructure to support current and future computing needs. It's flexible in content presentation and in providing a personalized experience for users according to their role or operational context.

Note
Starting March 31, 2022, NI SP EF Portal doesn't support VNC®, HP® RGS, VirtualGL, and DCV 2016 and previous versions. Integrations still might be working but are not supported.

Note
Starting August 5, 2022, NI SP EF Portal is bundled with an embedded version of Tomcat® that doesn't provide any sample webapps. According to CVE-2022-34305, the sample webapps are affected by a vulnerability. Without these sample webapps, EF Portal is not affected by the CVE-2022-34305 vulnerability.

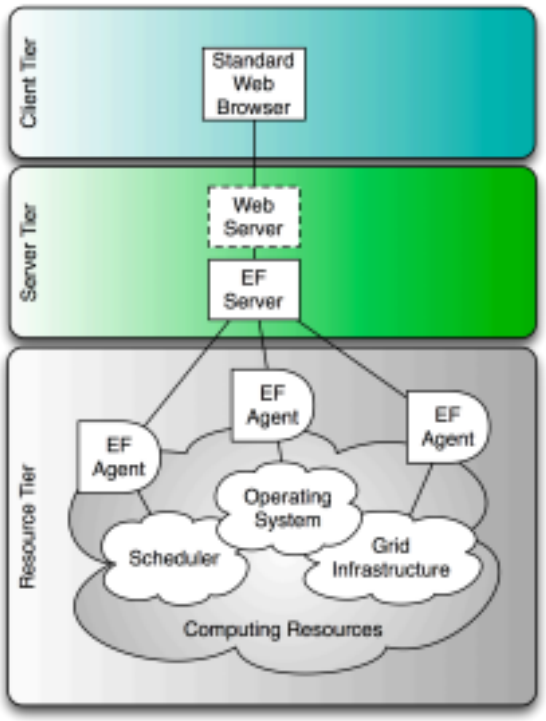
Architectural overview

EF Portal has an architecture layered into three tiers, as shown in EF Portal architecture:

- The *Client Tier* usually consists of the user's web browser. It provides an intuitive software interface that uses established web standards such as XHTML and JavaScript®. This tier is independent from the specific software and hardware environment that the end user uses. The Client Tier can also integrate remote visualization technologies such as DCV.
- The *Server Tier* consists of a *Server* that interacts with EF Portal Agents and manages the interaction with users.
- The *Resource Tier* consists of one or more *Agents* that are deployed on the back-end infrastructure. Agents manage computing resources on user's behalf and interact with the underlying operating system, job scheduler, or grid infrastructure to run EF Portal services. For example, they start jobs, move data, and retrieve cluster loads.

Topics

EF Portal Architecture



Basic workflow

EF Portal abstracts computing resources and data management (*Resource Tier*) and exposes services to users (*Server Tier*). Users access the services directly from their browsers (*Client Tier*).

The internal structure of EF Portal reflects this high-level architecture and revolves around two main software components: the EF Portal Server and the EF Portal Agent.

The EF Portal Server

> The EF Portal Server is a Java™ web application. It is deployed inside a Java Servlet container and exposes services to users. The EF Portal ships with Apache Tomcat®.

The EF Portal Agent

> The EF Portal Agent is a stand-alone Java™ application that manages the computing resources and runs services on user's behalf when running as root.

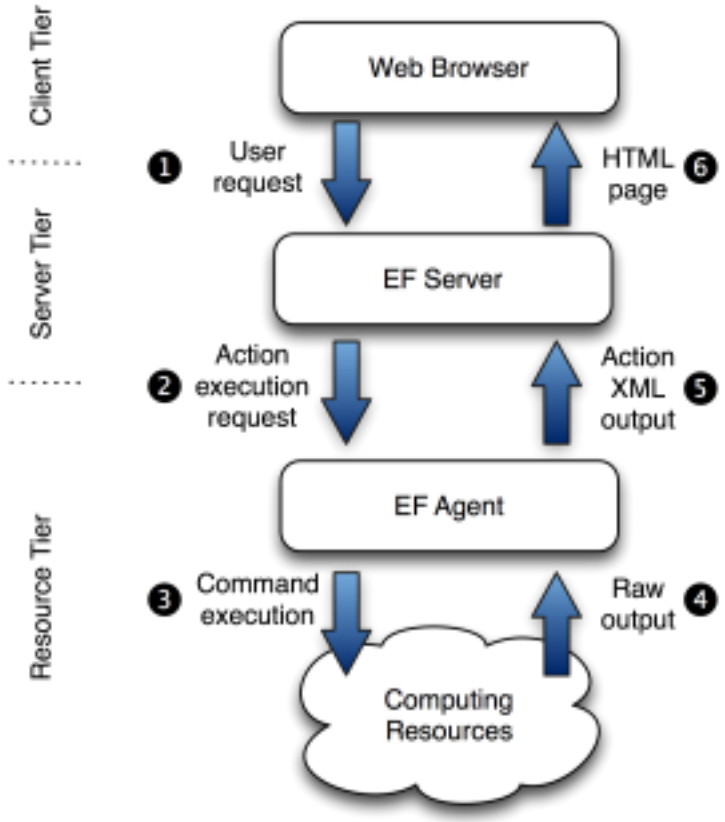
EF Portal Server receives incoming requests from a web browser. The web browser authenticates and authorizes them, and then asks an EF Portal Agent to run the required actions. For a visual illustration, see Interaction diagram.

Agents can perform different kinds of actions. This includes running a command on the underlying operating system and submitting a job on the grid infrastructure.

The results of the action that was run are gathered by the Agent and sent back to the Server.

The Server applies some post processing transformations, filters output according to defined access control lists (ACL), and transforms the results into an HTML page.

Interaction diagram



EF Portal creates or reuses a data area each time a new action is run. This area is called *spooler*. The spooler is the working directory of the action. It contains files uploaded when the action is submitted. Users can only download files from their spoolers.

The spooler is located on a file-system readable and writable by both Server and Agent. For more information, see Managing spoolers.

Deploying the service

When EF Portal Portal is installed on one host, it's called a *basic installation*. The Server Tier contains the Agent that's used to access the Resource Tier. The efnobody user runs the Server in this scenario.

This is the default user that you choose when you install EF Portal. The EF Portal Server contains a *local Agent* that's used when configured in your service description and you are submitting a scriptlet. For a visual illustration, see EF Portal deployed on one host .

In most cases, the Server contacts the default *remote Agent* that was configured during setup. Usually, the remote Agent runs as root and can do the following:

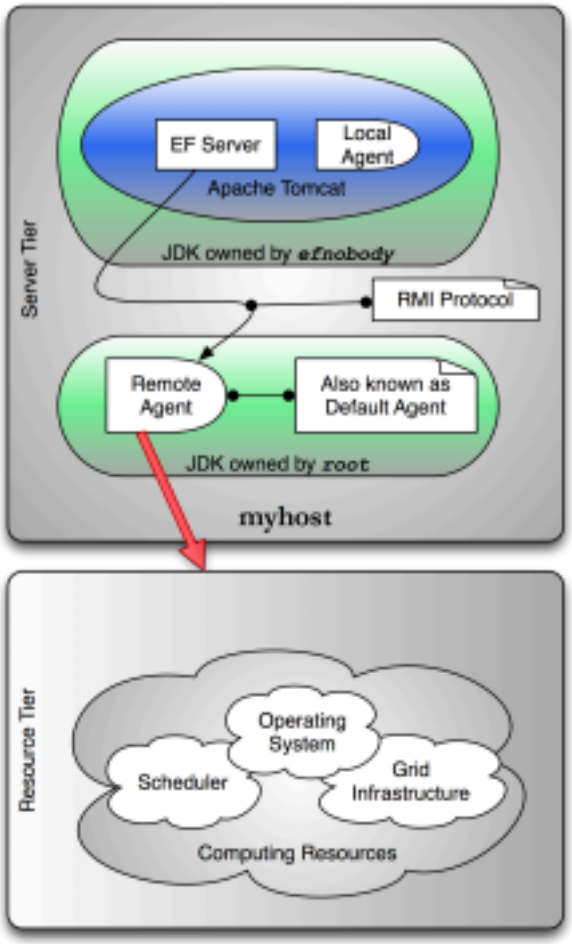
- Authenticate users using PAM/NIS.
- Create or delete spoolers on the user's behalf.
- Run services on the user's behalf.
- Download files on the user's behalf.

The remote Agent can also run as a user without permission. However, you lose the main features of an Agent running as root. All spoolers and services are created or run as this user without

permission. It also implies that EF Portal has to use an authentication module that doesn't require root privileges to check credentials, such as *LDAP* and *Active Directory*.

The Server communicates with the Remote Agent using Java™ RMI protocol. However, the local Agent is reached directly because it's inside the Java Virtual Machine (JVM) space in the Server.

EF Portal deployed on one host



Distributed deployment

EF Portal Server can be deployed in a demilitarized zone (DMZ) that's accessible from your intranet or the internet. The default EF Portal Agent resides in your protected computing environment. EF Portal Server and EF Portal Agent reside on different hosts in this scenario. This is called a *distributed deployment*. For a visual illustration, see EF Portal deployed on more hosts.

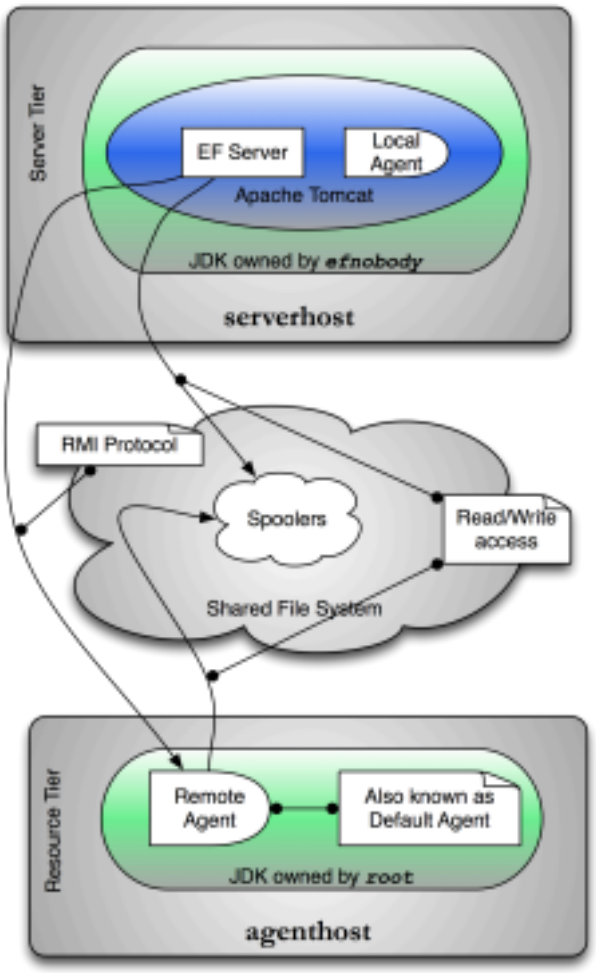
In this scenario, the following requirements must be met:

- The Server host reaches the Agent host on ports that were specified during setup.
- The Agent host reaches the Server host by HTTP on a port that was specified during setup.
- Spoolers are stored on a shared file-system.
- The Spooler shared file-system is readable and writable by both the efnobody and root users.
- The Server must reach the Agent using RMI. Otherwise, the user's submissions fail.
- The Agent must reach the Server using HTTP. Otherwise, the user's downloads fail.

Note
Spoolers must reside on a shared file-system for the following reasons. The Server saves files that

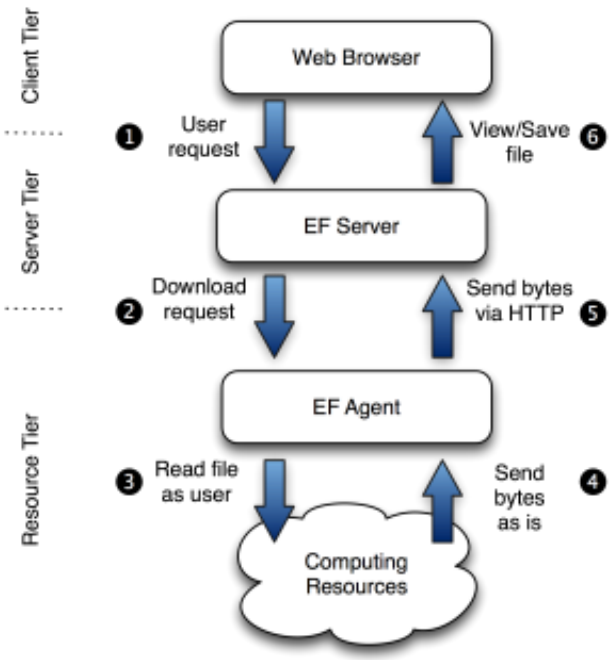
users send. A service runs on an Agent must access them. Because files are written by the Server, efnobody needs read access to traverse the directory structures when creating new spoolers and write access to write files. Because services are run on the user's behalf, root needs write permissions on the spoolers area to give directory and files ownership to the user that's running the service. This ownership change is necessary because the spooler and the files were created by efnobody.

EF Portal deployed on more hosts



File downloads

Users can download files located in their spoolers, their home directory, or any configured Places, depending on system settings. The diagram below illustrates the process flow.



1. EF Portal Server receives incoming requests from the Web Browser.
2. EF Portal Server forwards the request to EF Portal Agent, which then downloads the remote file.
3. As a user, EF Portal Agent forks a process which reads the file.
4. EF Portal Agent connects back to EF Portal Server using HTTP to send back the bytes that were produced by the forked process.
5. EF Portal Server sends the bytes back to the browser.
6. The browser displays the file or proposes to save it on disk depending on file mime-type and browser settings. For more information, see Managing internet media types.

Step 4 highlights why it's important for EF Portal Agent to reach EF Portal Server using HTTP.

You can use EF Portal to download files in *streaming* mode. File contents are displayed while they're being downloaded. This is useful for files that grow when the service is running. The flow is the same as the remote file download except that EF Portal Server polls, at fixed intervals, EF Portal Agent for some fresh data. This feature mimics the Unix **tail** that displays the last file portion while it grows.

Interactive session broker

EF Portal 2021.0 and following versions include the *interactive* plugin, a session broker that's scalable and reliable. Its main purpose is to ease application delivery and manage interactive sessions.

Deployment

The solution relies on the following systems:

- NI SP EF Portal. This is the kernel that Interactive Plugin is built on.
- A resource manager software that allocates and reserves resources according to the specified resource sharing policy or a session broker.
- One or more remote visualization middleware platforms, such as DCV.

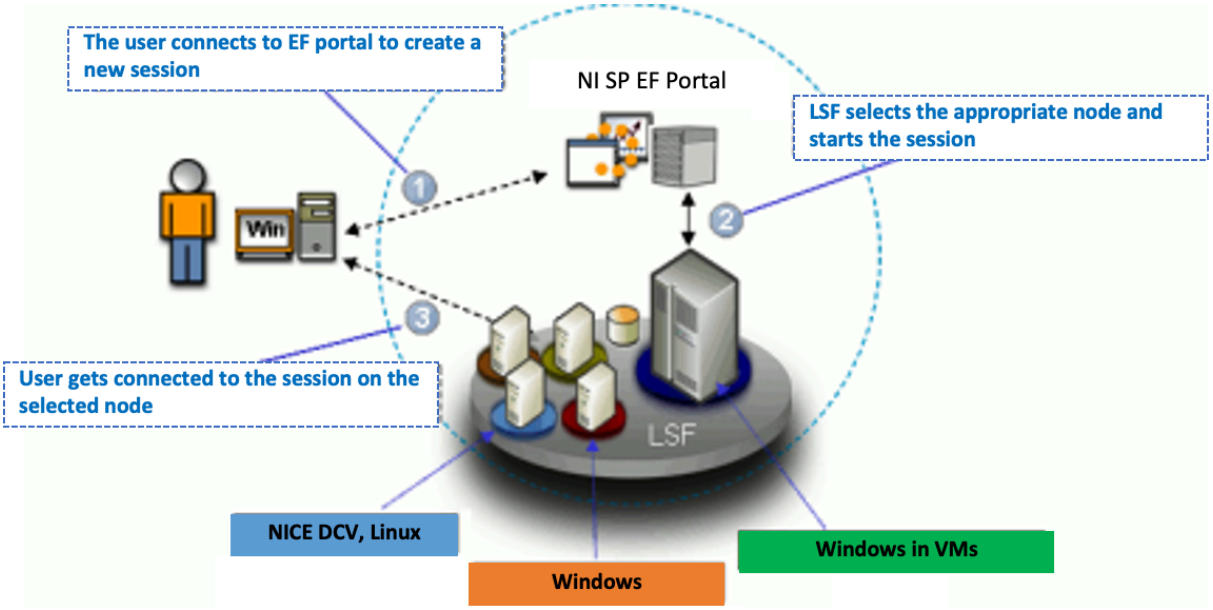
For a complete list of the supported HPC workload managers, session brokers, and visualization middleware, see Prerequisites.

The visualization farm can be Linux® or Windows®.

Workflow

The following picture represents a real-world example of infrastructure, including nodes with DCV. You can use this infrastructure to deliver 2D and 3D applications on Windows® and Linux® through DCV.

Interactive Plugin Use Model



The following explains what happens at each step:

1. The user connects to Interactive Plugin to create a new session. Each session is a distinct job of the underlying HPC workload scheduler or a distinct session in the underlying third-party session broker.
2. The resource manager or the session broker schedules the new session on the most appropriate node. This node complies with the application requirements and the resource sharing policies.
3. After the session is created, Interactive Plugin sends a file to the web browser. This file contains information that allows the browser to start the correct visualization client. The client uses this information to connect to the remote session.

EF Portal plugins

A plugin is a piece of software that extends EF Portal Portal. NI SP sells and provides many of these extensions at no cost.

The plugins can extend EF Portal in many different areas:

- **Bundle** - a full-featured package containing other plug-ins.
- **Kernel** - an extension that enhances EF Portal core system (for example, *REST, Interactive Plug-in*).
- **Auth** - an extension that authenticates users against an authoritative source (for example, *PAM Plug in*).
- **Data** - an extension that helps display data inside EF Portal Portal (for example, *File*

Manager, RSpooler Plug-in).

- **Grid** - an extension that connects EF Portal Portal with a grid manager (for example, *LSF Plug-in*).
- **Util** - additional utility components (for example, *Demo Portal*).

NI SP ships many plugins according to these conventions:

- **Certified extensions** - are developed and supported by NI SP. They're available and supported as add on products. Add-on products pass a quality assurance process at every new release of EF Portal. Each extension is individually certified to work on the latest release of EF Portal. The guidelines are provided to evaluate how different groups of extensions might interact. No implicit commitment is taken about the compatibility between two different extensions.
- **Qualified extensions** - are developed or modified by NI SP. This ensures professional development and good functionality under some specific EF Portal configuration. Qualified extensions are available as project-accelerator solutions, to facilitate integration of your EF Portal Portal in specific complex scenarios. Further support can be provided as Professional Services.
- **Contributed extensions** - are developed by third parties and are made available by the respective authors. They're provided as-is, and no additional endorsement is provided by NI SP. Further support on such modules might be sought from the contributing authors, if available.

EF Portal Enterprise

This section describes the EF Portal Enterprise version, the solution aimed at enterprise environments where *load balancing* and *fault tolerance* are crucial requirements.

All the general concepts about EF Portal explained in the previous sections apply also to the EF Portal Enterprise version. The following sections illustrate the characteristics of the Enterprise solution. They describe the architecture, highlight the differences with the architectures described earlier, and suggest the best approach for deployment.

Topics

Architecture

EF Portal Enterprise architecture involves multiple EF Portal Servers and multiple EF Portal Agents. All the Servers and the Agents maintain the same role and functionalities. However, they do so in an EF Portal Enterprise infrastructure and EF Portal Servers are able to communicate with each other over the network to share and manage the system status.

The shared system status involves the following resources:

- The users' spoolers and spoolers repository
- EF Portal triggers
- Users that are logged in
- EF Portal license tokens

Information is shared among EF Portal Servers and managed in a distributed architecture where there's no single point of failure in the system. Each of the servers alone can cover all the needed

functionalities and, at the occurrence, it can keep the whole system up and running, making the system more robust and fault tolerant.

The EF Portal Enterprise solution relies on a file-system that's not only shared between an EF Portal Server and an EF Portal Agent. It's shared among all the Servers and Agents. The EF Portal Agents need access to the spoolers area. However, EF Portal Servers have stronger requirements and need other file-system resources to be shared besides spoolers. These include the EF Portal repository files that contain server-side metadata about spoolers, the file upload cache, and the plugins data directory tree. For more information about the suggested and supported approach for file-system sharing, see Deployment.

Another important component to consider in the EF Portal architecture is the Database Management System (DBMS). In a standard EF Portal installation, you can rely on the Apache Derby® database distributed with EF Portal. In the EF Portal Enterprise solution, however, use an external JDBC compliant DBMS. All the EF Portal Servers must have access to the database. For a list of the supported DBMS, see Database management systems.

To maintain a single point of access to EF Portal, the architecture involves a front-end HTTP/S network load balancer. This component isn't part of the EF Portal Enterprise deployment. Rather, it's a third-party solution, which might be either a software or hardware component. This might be, for example, a Cisco router of the 6500 or 7600 series. The router, in this example, is configured with the sticky session capability † that dispatches users' requests to the EF Portal Servers in a balanced way.

NI SP can provide and set up the network load balancer based on third-party technology, such as Apache® Web server, as professional services activity according to specific projects with customers.

† Sticky session refers to the feature of many commercial load balancing solutions for web-farms to route the requests for a particular session to the same physical machine that serviced the first request for that session. The balancing occurs on web sessions. It doesn't occur on the single received web requests.

Software distribution and license

EF Portal Enterprise is distributed with the *same software package* of EF Portal. It's the EF Portal software license that specifically enables EF Portal Enterprise capabilities. EF Portal doesn't need a license on an EC2 instance. For instructions on how to get the EF Portal software and license, see Obtaining NI SP EF Portal.

The following is an example of an EF Portal Enterprise license.

```
<?xml version="1.0"?>
<ef-licenses>
 <ef-license-group product="EF Portal HPC ENT" release="2024.0"
format="2">
   <ef-license
 component="EF Base"
 vendor="NI SP Software"
 expiration="2015-12-31"
 ip="172.16.10.171,172.16.10.172"
 licensee="NI SP RnD Team"
 type="DEMO"
 units="100"
 units-per-user="1"
 license-hosts="false"
 hosts-preemption="false"
 signature="MC0CFQCGPmb31gpiGxxEr0DdyoYud..." <!-- Omitted -->
```

```
 />
 </ef-license-group>
</ef-licenses>
```

The product attribute value is EF Portal HPC ENT. This value defines an EF Portal license for HPC environments with the *ENT* string specifying the Enterprise version. The ip attribute of tag ef license with the list of the IP addresses of the licensed EF Portal Servers nodes.

Deployment

Because of its distributed architecture, deployment scenarios of EF Portal Enterprise might be different and vary in complexity.

You can have each component (specifically EF Portal Server and Agent) on a different node. You can also decide to pick only the minimum number of parts of the file system to share on each of the nodes. Remember that, in an EF Portal Enterprise deployment, you also have file-system resources to be shared among EF Portal Servers. In many cases, even when resources don't necessarily require sharing, make sure that they're replicated and maintained in alignment among EF Portal Servers.

Even if, in principle, it's possible to fine-tune the installation of EF Portal Enterprise, consider the different factors, such as networking and file-system sharing. It's common practice to go for the comparatively fast and easy-to-maintain deployment approach that's described here. If this one doesn't meet your specific requirements, you can use a different approach. Discuss your requirements with NI SP professional services to see what is the best approach for you.

The suggested approach to deploy EF Portal Enterprise involves the following:

- One node for each pair of EF Portal Server and Agent that you want to install.
- A shared file-system for the whole `$EF_TOP` directory tree.
- `$EF_TOP` is the top EF Portal installation directory.

For more information, see Installation directories.

With this approach, you can install and manage the software from one node. All the binaries and data directories such as spoolers, sessions, and licenses are shared among the installation nodes.

For those resources that are expected to be local but might conflict in a shared environment, EF Portal provides a per-hostname directory tree. This directory tree includes a shared environment like the logging directory where each Server and Agent writes log files with the same names.

We recommend that you host an external database management system (DBMS) on different nodes and configure them to be fault tolerant.

When you install EF Portal Enterprise, you insert the JDBC URL in the EF Portal database instance together with the username and password that you use to access it. The EF Portal database instance must be previously created empty, EF Portal creates all the needed tables the first time you connect to it.

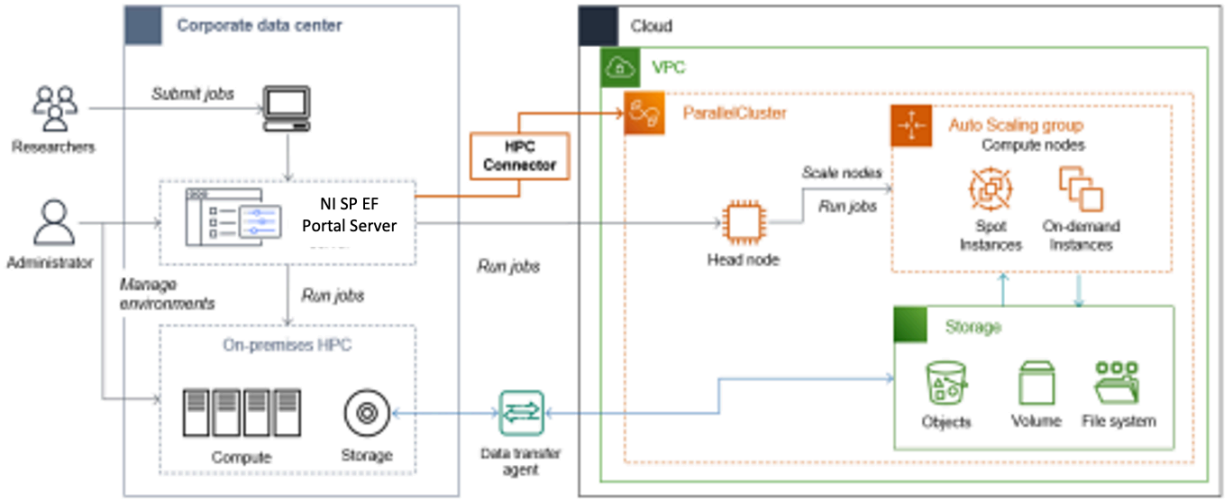
The details that specifically concern an EF Portal Enterprise deployment, its requirements and installation notes, are integrated where needed in the next chapters of this guide.

AWS HPC Connector

AWS HPC Connector integration is kept as-is and is not supported.

HPC Connector provides a straight-forward way for HPC customers to use the elastic infrastructure that AWS ParallelCluster dynamically creates and manages on AWS. You can

choose to install EF Portal on-premises and use HPC Connector to extend available HPC environments to AWS. This installation scenario is suitable for use cases such as cloud bursting. Or, you can install EF Portal directly on an Amazon EC2 instance and use HPC Connector to run workflows entirely on AWS. HPC Connector uses customer-defined cluster configurations for creating clusters in the cloud that you can use for running jobs.



You can use HPC Connector flexibly as your needs change over time. For example, you can use it to experiment with running select workloads on AWS or to manage most or all of your workloads on AWS. You can use HPC Connector to manage your HPC workloads from both your on-premises and AWS environments in a centralized fashion. For example, you might be migrating your workloads to AWS and need to manage and maintain both environments for some time. Or, you want to burst some workloads to the cloud when on-premises resources are insufficient to meet your requirements. Having access to elastic AWS resources can help to increase the productivity of your researchers. Moreover, you can use HPC Connector to get started managing hybrid on-premises and AWS environments in a more centralized manner.

Topics

How HPC Connector works

HPC Connector works by using AWS ParallelCluster. AWS ParallelCluster is an open-source cluster management tool on AWS that you can use to deploy and manage HPC clusters on AWS. It uses a simple text file to model and provision all the resources needed for your HPC applications in an automated and secure manner. With AWS ParallelCluster, the resources needed for your applications are dynamically scaled in an automated and secure manner. After a burst of jobs is completed, AWS ParallelCluster terminates the instances it created. This leaves only the head node and the minimum capacity defined active and ready to scale up new compute nodes when required.

When submitting a job to a remote cluster hosted on AWS , HPC Connector relies on AWS Systems Manager Session Manager for the job submission process, and uses Amazon S3 for transferring local data to the node running EF Portal server through and from the remote cluster. In order to do that, HPC Connector requests temporary credentials to AWS Identity and Access Management (IAM) for managing the resources on AWS . When you submit a job to a remote cluster, HPC Connector performs the following activities.

- Creates an Amazon S3 folder for transferring data from the user's local spooler.

- Generates a dynamic policy to restrict Amazon S3 access to the newly created Amazon S3 folder.
- Creates a new set of temporary credentials that use the generated policy.
- Launches a script for transferring the data that's in the local spooler to S3 using such credentials.
- Runs a remote script on the head node of the remote cluster that uses AWS SSM, which performs the following actions.
- Retrieves the data from the S3 folder using a set of temporary credentials (using the same mechanism described above). The data is then copied to the destination folder.
- Changes the files and folder permissions to give ownership of the files to the remote user.
- Submits the job as a remote user from within the destination folder, with the proper environment variables and launching the job script associated with the EF Portal service.

HPC Connector run requirements

For HPC Connector to work properly, you must have an AWS account and create and configure the following items.

- An Amazon S3 bucket that's used by HPC Connector for transferring the data from the spooler local to the node running the EF Portal server to the remote clusters on AWS, or the other way around. HPC Connector doesn't support Using Amazon S3 bucket keys.
- An IAM role for accessing the Amazon S3 bucket. This role is used by HPC Connector for transferring the data back and forth to the remote destination.
- An IAM role for managing AWS ParallelCluster in your account. This role is used by HPC Connector for creating, starting, and stopping clusters.
- An IAM role for running jobs remotely using AWS SSM. This role is used for launching job scripts remotely on the clusters.
- For EF Portal on-premises, an IAM user for allowing HPC Connector to assume the required roles when managing clusters or launching remote jobs.

Cluster users

When launching a cluster from within an HPC Connector, administrators must specify how to map users within the cluster with respect to the user launching the job. HPC Connector supports the following two different mechanisms for this.

- **Single user mode** – In this mode, any user connected to the EF Portal portal and with access to the cluster submits jobs to the cluster as a single predefined cluster user. This mode requires administrators to specify the user name to use when creating the cluster (for example, ec2-user or ubuntu).
- **Multi-user 1:1 mode** – In this mode, any user connected to the EF Portal portal and with access to the cluster submits jobs to the cluster as a remote cluster user with the same name.

HPC Connector does not provision or manage users on the cluster. This means that you're required to have such users already present in the cluster before submitting any job. Additionally, you might also need a mechanism to keep users in sync with your on-premises environment. HPC Connector maps the on-premises and remote users with the chosen logic. However, if the chosen user isn't available in the remote cluster, your job submissions will fail.

When requested to launch a job remotely, HPC Connector launches the job using AWS SSM. HPC Connector assumes the role of a remote user on the cluster depending on the mode. This is either the user specified when the cluster is created for single user mode or a user with the same name as the one logged in to the EF Portal portal for multi-user mode.

Obtaining NI SP EF Portal

If you didn't already receive your NI SP EF Portal package from NI SP or your EF Portal reseller, download it from the EF Portal website.

Downloading EF Portal

EF Portal packages can be downloaded from the NI SP EF Portal website:

https://www.ni-sp-software.com/download/

Contact <support@ni-sp-software.com> or your EF Portal reseller for any questions.

EF Portal on Amazon EC2

You don't need a license server to install and use the EF Portal server on an Amazon EC2 instance. The EF Portal server automatically detects that it's running on an Amazon EC2 instance. It also periodically connects to an Amazon S3 bucket to determine if a valid license is available. Make sure that your instance can do the following:

- It can reach the Amazon S3 endpoint. If it has access to the internet, it connects using the Amazon S3 public endpoint. If your instance doesn't have access to the internet, configure a gateway endpoint for your VPC with an outbound security group rule. Alternatively, configure it with an access control list (ACL) policy that allows you to reach Amazon S3 through HTTPS. For more information, see Gateway VPC Endpoints in the Amazon VPC User Guide. If you experience any issues connecting to the S3 bucket, see Why can't I connect to an S3 bucket using a Gateway VPC endpoint? in the AWS Knowledge Center.
- It has permission to access the required Amazon S3 object. Add the following Amazon S3 access policy to the instance's IAM role and replace the Region placeholder (*region*) with your AWS Region (for example, us-east-1). For more information, see Create IAM Role.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "s3:GetObject",
 "Resource": "arn:aws:s3:::efportal-license.region/*"
 }
]
}
```

Access to the instance metadata must be enabled. By default, it's enabled unless you might have turned it off. You can turn it back on by using the modify-instance-metadata-options command.

If you're installing and using the EF Portal server on an Amazon EC2 instance, you can skip the rest of this chapter. The rest of this chapter only applies to using the EF Portal server on an on-premises server or one hosted in the cloud.

EF Portal on on-premises and other cloud-based servers

When running on an on-premises server or one hosted in the cloud, you need a valid license to install and run EF Portal. If you don't already have one, contact <support@ni-sp-software.com> or your EF Portal reseller.

EF Portal licenses are classified as one of the following types:

- *Demo licenses* - demo licenses aren't bound to any IP address and are valid for one month.
- *Full licenses* - full licenses have time-unlimited validity and are bound to one or more IP addresses.
- *Year licenses* - year licenses have time-limited validity and are bound to one or more IP addresses.

Contact <support@ni-sp-software.com> or your EF Portal reseller to purchase, renew, or update a license. They can also help you to change your license or obtain a demo license.

Licensed plug-ins

When running EF Portal on Amazon EC2, a license plugin isn't required. When running EF Portal on an on-premises server or one hosted in the cloud, plug-ins require a specific license to work. The standard plug-ins that are included in the EF Portal installation. This requires that a license have the following characteristics:

- **interactive** - Enables basic functionalities for Interactive Session management.
- **applications** - Enables the Workspace.
- **hpc-support** - Enables the HPC functionalities.

Additional plug-ins provided by NI SP might also require a license.

The license file that's provided by your sales contact in most cases contains license components for all the plug-ins that are included in your EF Portal bundle.

For more information, check with our support support@ni-sp-software.com or your NI SP sales contact.

If you have a specific license file, it must be copied under `$EF_TOP/license` folder, and have an *.ef* extension.

It's automatically read by the portal. You don't need to restart the EF Portal.

Note
Remove from the `$EF_TOP/license` folder any older *.ef* license files that contain expired licenses or licenses that aren't correct. You want to do this because EF Portal doesn't accept any license conflict.

The following is an example license for the *Interactive* component.

```
<?xml version="1.0"?>

<ef-licenses>
<ef-license-group product="EF Portal PRO" format="1.0"
release="2024.0">
   <ef-license
 component="interactive"
 vendor="NI SP Software"
 expiration="2014-12-31"
 ip="10.20.10.14"
 licensee="Acme.com"
 type="DEMO"
 units="20"
 signature="xxxxxx"
 />
</ef-license-group>
</ef-licenses>
```

For more information about EF Portal licenses, see EF Portal licenses.

Planning a NI SP EF Portal deployment

Setting up EF Portal is a straightforward process. However, it's important to accurately plan your EF Portal Portal deployment to achieve seamless integration with your computing environment and to meet the IT requirements of your organization.

Note
Starting March 31, 2022, NI SP EF Portal doesn't support VNC®, HP® RGS, VirtualGL, and DCV 2016 and previous versions. Integrations still might be working but are not supported.

Prerequisites

Before you deploy EF Portal Portal, make sure that your system meets the following requirements.

Topics

- System requirements
- Third-party software prerequisites
- Network requirements
- Supported browsers
- Interactive Plugin requirements
- EF Portal Enterprise system requirements

System requirements

Topics

- Additional considerations for using SUSE Linux

NI SP EF Portal supports the following operating systems:

- Amazon™ Linux® 2, Amazon Linux 2023
- Red Hat® Enterprise Linux®, CentOS, Rocky, AlmaLinux 7.x, 8.x, 9.x (*x86-64*)
- Ubuntu 20, 22, 24 (*x86-64*)
- SUSE® Linux® Enterprise Server 12 SP5 (*x86-64*), SUSE® Linux® Enterprise Server 15 SP2 (*x86-64*)

Note
Other Linux® distributions and compatible Java™ versions might work but are not officially supported. Contact <support@ni-sp-software.com> for more information.

The installation machine must have at least 3 GB of RAM and one or more IP addresses. For these IP addresses, at least one of them must be reachable by each of the potential client machines. It can be reached either directly or through proxies.

To install EF Portal, minimally you need at least 200 MB of free disk space. However, we recommend that you have as much as 2 GB or more. This is because EF Portal while operating saves important data and logging information.

Make sure you have enough space for the service data that's stored inside the EF Portal spoolers. By default, spoolers are located inside the EF Portal installation directory ($EF_TOP/spoolers).

Additional considerations for using SUSE Linux

EF Portal PAM standard user authentication (system) expects to find the file `system-auth` in the folder `/etc/pam.d/`. However, in SUSE® Linux® Enterprise Server, this file is called `common-auth`. So, to make the standard authentication work, a symbolic link is required:

```
ln -s /etc/pam.d/common auth /etc/pam.d/system-auth
```

Third-party software prerequisites

In addition to the standard packages that are installed with your operating system, NI SP EF Portal also requires some additional third-party software. This topic describes the third-party software that's required and how you can set it up.

Topics

Java™ platform

NI SP EF Portal requires the *Linux® x64* version of *Oracle® Java™ Platform Standard Edition* (Java™ SE) or the *OpenJDK Runtime Environment* . EF Portal supports version 11 of these packages.

Supported Java™ vendors:

- Oracle®
- Amazon Web Services
- Red Hat®
- IcedTea

In this topic, JAVA_HOME is referred to as the Java™ installation directory.

The same Java™ version must be used for both EF Portal Server and EF Portal Agent.

Support for Java™ 8 has been discontinued starting with EF Portal 2026.0.

Database management systems

EF Portal requires a *JDBC-compliant* database. EF Portal uses a relational database management system to manage *Triggers*, *Job-Cache*, and *Applications* and *Views* user groups. EF Portal *Triggers* rely on Quartz (http://www.quartz-scheduler.org) engine to schedule EF Portal services to run. Triggers are used internally to run periodic tasks to check and update Interactive sessions status. They're also used to collect EF Portal usage statistics. The *Job-Cache* feature is responsible for collecting and caching job statuses over time.

By default, Apache Derby® database is installed together with EF Portal Professional.

Apache Derby® isn't supported for EF Portal Enterprise installations. We recommend that you use an external JDBC-compliant relational database management system (RDBMS). Because EF Portal Enterprise is part of a high availability solution, the RDBMS that you choose to use must have its own high availability strategy. We recommend that you configure the external RDBMS on a different node or nodes than the EF Portal servers. If possible, we also recommend that you configure it to be fault tolerant.

EF Portal supports MySQL® Database 8.0.x and later with the InnoDB storage engine. You can also use EF Portal with other databases, such as Oracle® Database, SQL Server®, MariaDB®. If you encounter issues with a supported relational database management system version, contact support@ni-sp software.com.

EF Portal provides the JDBC driver only for Apache Derby®. If a different DBMS is used, you must add the JDBC driver to the `$EF_TOP/<VERSION>/enginframe/WEBAPP/WEB-INF/lib` directory after you install EF Portal.

For instructions on how to install and configure the JDBC driver, see Install and configure EF Portal.

Authentication methods

You can use a variety of authentication methods with EF Portal. Some of them require third-party software components.

We recommend that you consider a variety of factors when choosing an authentication method. The following table details several relevant considerations including third-party software prerequisites, if any exist. For more information, see also Supported authentication methods.

Supported authentication methods

Name	Prerequisites	Notes
PAM	Linux® PAM must be correctly configured	This is the most common authentication method. As a system administrator, you can use it to add new authentication methods by installing new PAM modules. You can also use it to modify authentication policies by editing the configuration files. When you install it, you're asked to specify which PAM service to use, **system-auth** (or **common-auth**) is the default.

Name	Prerequisites	Notes
LDAP Active Directory	The **ldapsearch** command must be installed and working appropriately on the EF Portal Agent host.	You can use these authentication methods to authenticate users against an LDAP or Active Directory server. When you install these authentication methods, the EF Portal installer asks you to specify the parameters that are required by **ldapsearch** to contact and query your directory server.
HTTP Authentication	External HTTP authentication system	This authentication method relies on an external authentication system to authenticate the users. The external system then adds an HTTP authentication header to the user requests. EF Portal trusts the HTTP authentication header.
Certificate	SSL Certificates must be installed and exchanged between EF Portal Server and clients.	This authentication method is accomplished on the web server, which requires the client authentication through the use of SSL certificates.

The EF Portal installer can optionally verify if you configured the selected authentication method.

NI SP EF Portal can be easily extended to add support for custom authentication mechanisms.

Distributed resource managers

EF Portal supports different distributed resource managers (DRM).

When you install EF Portal, you must specify which distributed resource managers that you want to use and provide the information that's required by EF Portal to contact them. A single EF Portal instance can access more than one DRM at the same time.

Supported distributed resource managers

Name	Version	Notes
SLURM™	19.05.x - 25.0.x	SLURM™ binaries must be installed on the EF Portal Server host. SLURM™ master host must be reachable from the EF Portal Server host. The installer asks you to specify the path where binaries are installed. For SLURM configuration, specifically related to compute nodes that are dedicated to interactive sessions, the Feature dcv2 and RealMemory parameters must be added to every required node. 'dcv2' stands for DCV and is optional starting from EFP 2026.0.
IBM® Platform™ LSF®	10.1.x	The LSF client software must be installed on the EF Portal Agent host. The installer asks you to specify the LSF profile file.
Altair® PBS Professional®, OpenPBS®	Altair® PBS Professional®: 19.2.x - 2020.1.x OpenPBS®: 19.1.x - 23.06.x	The OpenPBS® or PBS Professional® client software must be installed on the EF Portal Agent host. The installer asks you to specify the directory where the OpenPBS® or PBS Professional® client software is installed.
TORQUE Resource Manager	6.0.x - 7.0.x	The TORQUE client software must be installed on the EF Portal Agent host. The installer asks you to specify the directory where the TORQUE client software is installed.

Name	Version	Notes
Sun® Grid Engine (SGE)	8.1.x	The Grid Engine client software must be installed on the EF Portal Agent host. The `$SGE_ROOT/$SGE_CELL/common` must be shared from SGE master to EF nodes. The installer asks you to specify the Grid Engine shell settings file.
Univa® Grid Engine® (UGE)	8.6.x	
Son of Grid Engine (SoGE)	8.1.x	
Gridware	9.x	The Gridware client software must be installed on the EF Portal Agent host.

As-is and unsupported integration of distributed resource managers

AWS Batch	The AWS Batch cluster must be created with AWS ParallelCluster 3.x (3.0.0 or later) and AWS ParallelCluster Batch CLI 1.0.0 must be installed.	The installer asks you to specify the AWS ParallelCluster cluster name, the AWS Region, the local AWS profile, and the IAM role to use to interact with AWS.
AWS HPC Connector	AWS HPC Connector requires AWS ParallelCluster version 3.0.2 or later.	The installer asks you to specify the AWS account ID, the AWS Region, the S3 bucket, and the IAM roles to use. Not supported. Integration kept as is.

By default, some schedulers such as PBS Professional® and Univa® Grid Engine® (UGE) 8.2.0 have job history disabled. This means that a job disappears when finished. We recommend that you configure these distributed resource managers to retain information about the finished jobs. For more information, see Required DRM configuration.

Note
Starting March 31, 2022, NI SP EF Portal doesn't support VNC®, HP® RGS, VirtualGL, and DCV 2016 and previous versions. Integrations are still available but not supported.

Required DRM configuration

Altair® PBS Professional®

Applies to versions: 11, 12, 14

By default, Altair® PBS Professional® doesn't show finished jobs. To enable job history, a server parameter must be changed:

```
qmgr -c "set server job_history_enable = True"
```

After it's enabled, the default duration of the job history is 2 weeks.

Univa® Grid Engine® (UGE)

Applies to versions: 8.2.x.

Univa® Grid Engine® (UGE) by default does not show finished jobs. To enable job history, follow these steps:

- *(8.2.0 only)* disable reader threads:

  Edit file `SGE_ROOT/SGE_CELL/common/bootstrap`.

  Set reader_threads to 0 instead of 2.
- Enable finished jobs:

  Run the `qconf -mconf` command.

  Set finished_jobs to a non-zero value according to the rate of finishing jobs.

  The finished_jobs parameter defines the number of finished jobs stored. If this maximum number is reached, the oldest finished job is discarded for every new job that's added to the finished job list.

  By default, EF Portal grabs the scheduler jobs every minute. The finished_jobs parameter must be tweaked so that a finished job stays in the job list for at least a minute. Depending on the number of jobs that are running in the cluster a reasonable value is in between *the medium number of running jobs* and *the amount of jobs ending per minute*.
- Run the `restart qmaster` command.

SLURM™

Applies to versions: all.

SLURM™ shows finished jobs for a default period that's defined by the `MinJobAge` parameter in the slurm.conf file. It's under `/etc/slurm` or the SLURM™ configuration directory. The default value is *300* seconds (five minutes).

If you changed this parameter, ensure it's not set to a value lower than 300.

Check the `MaxJobCount` parameter isn't set.

After changing this parameter, restart SLURM™ by running the `/etc/init.d/slurm stop; /etc/init.d/slurm start` command.

This must be set on all SLURM™ nodes.

IBM® Platform™ LSF®

Applies to versions: all.

IBM® Platform™ LSF® shows finished jobs for a default period that's defined by the `CLEAN_PERIOD` parameter in the `lsb.params` file. The default value is 3600 seconds (one hour).

If you changed this parameter, ensure it's not set to a value lower than 300.

After changing this parameter, run the `badmin reconfig` command.

AWS HPC Connector

For information, see Requirements for running HPC Connector.

AWS Batch

To integrate EF Portal with AWS Batch, create an AWS Batch cluster with AWS ParallelCluster and give the user that's to run the EF Portal server permission to interact with the cluster. To do this, follow these steps.

- Install AWS ParallelCluster and configure it. For more information, see Setting up AWS ParallelCluster in the *AWS ParallelCluster User Guide*.

As a part of the setup process, AWS CLI is installed as a dependency of AWS ParallelCluster.

- Install AWS ParallelCluster AWS Batch CLI 1.0.0. It's distributed on PyPi as aws parallelcluster-awsbatch-cli.
- Create a new cluster for the AWS Batch scheduler. Make sure that, when you create the cluster, you take into account the network requirements for AWS ParallelCluster with AWS Batch scheduler.
- In the IAM console create a user *MY_USER* with the following policy:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "sts:AssumeRole",
 "Resource": "<MY_ROLE_ARN>"
 }
 ]
}
```

For this policy, replace *<MY_ROLE_ARN>* with the ARN of the role that you create in the next step. Keep note of the User credentials because you're going to use them later.

• In the IAM console, create a role *MY_ROLE* that uses the policies "Base user policy" and "Additional user policy" when using AWS Batch scheduler". For more information, see AWS Identity and Access Management roles in AWS ParallelCluster in the *AWS ParallelCluster User Guide*. In addition, also include the following trust relationship.

```
{
 "Version":"2012-10-17",
 "Statement":[
 {
 "Effect":"Allow",
 "Action": "sts:AssumeRole",
 "Principal": {
 "AWS": "<MY_USER_ARN>"
 }
 }
 ]
}
```

Replace *MY_USER_ARN* with the ARN of the user that you created on the preceding step.

- Create a dedicated AWS profile with name *MY_PROFILE* to use with the AWS CLI for the

user running the EF Portal Server (for example, *efnobody*) and configure it to use the credentials of *MY_USER*.

```
[efnobody]$ aws configure --profile MY_PROFILE
```

- Follow EF Portal installer steps to configure AWS Batch EF Portal plugin to contact the created cluster.
- When upgrading to EF Portal 2024.0, add the following lines in $EF_CONF_ROOT/plugins/awsbatch/ef.awsbatch.conf config file after it's upgraded the release with the installer:

```
# AWS profile that the AWS Batch plugin should use
AWSBATCH_PROFILE=<MY_PROFILE>

# AWS IAM role ARN that the AWS Batch plugin should use to
interact with AWS Batch
AWSBATCH_ROLE_ARN=<MY_ROLE_ARN>
```

Replace *MY_PROFILE* and *MY_ROLE_ARN* with the specific ones that you created in the preceding steps.

Remote visualization technologies

EF Portal supports different remote visualization technologies, and the same EF Portal instance can manage more than one of these visualization technologies. The following table lists the supported ones.

Supported remote visualization technologies

Name	Version	Notes
DCV	2021.x or later	You can use it to share sessions both in full access or view only mode.
DCV Session Manager		For more information, see the DCV Session Manager documentation

For instructions on how to install and configure these remote visualization technologies, see their respective manuals.

Note
Starting March 2022, NI SP EF Portal doesn't support TurboVNC, in favor of DCV Session Manager. Customers using versions of EF Portal released after March 2022 will be unable to use TurboVNC for accessing interactive sessions. Integrations still might be working but are not supported.

Remote visualization technologies configuration

DCV on Linux

For Linux environments, the authentication configuration to use with DCV must correspond to the

authentication system that's set on the DCV server in the remote visualization hosts.

On EF Portal, the authentication to use with DCV on Linux can be set in the `INTERACTIVE_DEFAULT_DCV2_LINUX_AUTH` configuration parameter that's in the `$EF_TOP/conf/ plugins/interactive/interactive.efconf` file.

The default value and documentation can be found in the static configuration file that's named `$EF_TOP/<VERSION>/enginframe/plugins/interactive/conf/interactive.efconf`.

The *auto* authentication system provides seamless authentication with self-generated strong passwords. It requires the following configuration on the visualization hosts that are running the DCV server.

- Make sure that the DCV simple external authenticator that's provided with DCV is installed and running.

  The simple external authenticator installation package is distributed as an rpm (for example, nice dcv-simple-external-authenticator-2021.x...x86_64.rpm).

  After it's installed, you can manage the service as root user:
  - On systems using SystemD

```
$ systemctl [start|stop|status] dcvsimpleextauth
```

  - On systems using SysVInit

```
$ /etc/init.d/dcvsimpleextauth [start|stop|status]
```

- You must configure the DCV server to use the simple external authenticator instance that's named dcvsimpleextauth to run on the same host. For example, configure it to run by editing the `/etc/dcv/dcv.conf` file, under the security section as follows:

```
[security]
auth-token-verifier="http://localhost:8444"
```

- Restart the DCV server after changes were made to the /etc/dcv/dcv.conf configuration file.

DCV on Windows

For Windows environments the authentication configuration used with DCV must be configured on EF Portal in the `INTERACTIVE_DEFAULT_DCV2_WINDOWS_AUTH` configuration parameter. This is in the `$EF_TOP/conf/plugins/interactive/interactive.efconf` file.

`$EF_TOP/<VERSION>/enginframe/plugins/interactive/conf/interactive.efconf` static configuration file contains default values and documentation.

The auto authentication system provides seamless authentication with self-generated strong passwords. It doesn't require any other configuration on the visualization hosts that are running the DCV server.

The DCV server service is managed by the interactive session job landing on the node:

- If the DCV server service is not running, it's started.
- If the DCV server service is running but with a different authentication configuration than the one set on the EF Portal side, the configuration is changed and the service restarted. This is also the case if the DCV server is configured to launch the console session at system startup. This setting is removed by the interactive session job.

- If the DCV session is running but there's no logged user, the session is closed by the interactive session job.

Network requirements

EF Portal is a distributed system. Your network and firewall configuration must allow EF Portal components to communicate with each other and with user browsers.

The exact requirements depend on how EF Portal is deployed on your system. The table below summarizes the network requirements for a standard EF Portal installation.

Network requirements

Port (default)	Protocol	From host	To host	Mandatory
8080/8443	HTTP/HTTPS	User's clients	EF Portal Server	Mandatory
9999 and 9998	RMI (TCP)	EF Portal Server	EF Portal Agent	Optional
8080/8443	HTTP/HTTPS	EF Portal Agent	EF Portal Server	Optional †
7800	TCP	EF Portal Server	EF Portal Server	Mandatory only for EF Portal Enterprise ‡
3000	HTTP/HTTPS	User's clients	EF Portal Web Terminal	Optional
3001	HTTP/HTTPS	User's clients	EF Portal Proxy	Optional

† Required if EF Portal Agent and EF Portal Server run on separate hosts

‡ EF Portal Servers use the port to communicate with each other

Port customization

The HTTP/HTTPS server port is configured automatically during installation. You can change it manually by editing the `${EF_CONF_ROOT}/tomcat/conf/server.xml` file.

Two alternative `<Connector>` sections are available - one for HTTPS and one for HTTP. Only one of them must remain enabled:

```
<!-- HTTPS connector -->
<Connector port="8443" maxThreads="150" SSLEnabled="true"
           scheme="https" secure="true" clientAuth="false"
           sslProtocol="TLS"
keystoreFile="conf/certs/ef.tomcat.keystore"
           keystorePass="..."
           server="Apache" URIEncoding="utf-8"/>

<!-- HTTP connector -->
<Connector port="8080" maxHttpHeaderSize="8192"
           maxThreads="150" minSpareThreads="25"
           enableLookups="false" redirectPort="8443" acceptCount="100"
           connectionTimeout="20000" disableUploadTimeout="true"
           URIEncoding="utf-8" />
```

Switching between HTTP and HTTPS requires:

- Configuring certificates when enabling HTTPS
- Restarting the EF Portal Server

Starting from EF Portal 2026.0, RMI ports can be configured through the Web Configuration Editor by modifying the `ef.agent.port` and `ef.agent.bind.port` parameters. These parameters must be updated in both `agent.conf` and `server.conf`. A restart of both server and agent daemons is required.

Port 7800 is used for inter-server communication in EF Portal Enterprise only. It can be customized by editing the `${EF_ROOT}/enginframe/conf/hazelcast.xml` and by setting the corresponding value in the `ef.enterprise.servers` property within `server.conf`.

Supported browsers

NI SP EF Portal produces HTML that can be viewed with most popular browsers. NI SP EF Portal was tested with the browsers that are listed in Supported browsers.

Supported browsers

Name	Version	Notes
Microsoft® Edge	41 and 44 and later	
Mozilla Firefox®	3.6 and later	
Apple® Safari®	6.0 and later and iOS 6 version	Tested on Mac® OS X® and iPad® only.
Google™ Chrome™	25 and later	

JavaScript® and Cookies must be enabled on browsers.

At the end of December 2021, we discontinued support for Internet Explorer 10. At the end of June 2022, we discontinued support for Internet Explorer 11.

Interactive Plugin requirements

Interactive Plugin requires the following components to be successfully installed and configured:

- At least one supported resource manager software or a session manager. For more information, see Distributed resource managers and Session Managers.
- At least one supported remote visualization middleware. For more information, see Remote visualization technologies.

When running EF Portal on an Amazon EC2 instance, you can use the Interactive Plugin without a license. When running on an on-premises or alternative cloud-based server, you need a license that's installed on the EF Portal Server.

Make sure that each node that's running interactive sessions has all the necessary software installed. On Linux®, this usually means the packages for the desired desktop environment, such as gnome, kde, or xfce.

In addition, install the following software to make it available in the system PATH on visualization

nodes. When you do this, the portal can show screen thumbnails in the session list.

- Linux®: *ImageMagick tool* (http://www.imagemagick.org) and the xorg-x11-apps, xorg-x11-utils packages
- Windows®: *NICE Shot Tool* (`niceshot-uknown.exe`, available under `$EF_TOP/<VERSION>/enginframe/plugins/interactive/tools/niceshot`).

Session Manager

Starting from version 2020.0, EF Portal supports DCV Session Manager as Session Broker.

When you install EF Portal, you can choose to use DCV Session Manager as session broker. Then, provide the configuration parameters that are required by EF Portal to contact the remote DCV Session Manager Server.

Single application desktop requirements (Linux®)

In some workflows, you might want to run a minimal session on your interactive nodes consisting of a minimal desktop and a single application running. For this use case, instead of installing a full desktop environment such as GNOME or KDE, we recommend that you only install the required tools. The required tools are a Window manager, a dock panel, and any of the applications that you intend to use.

In this example scenario, you can configure the minimal.xstartup script to be a Window Manager choice for the Applications and Views service editors.

Here is a reference list of the tools that the `minimal.xstartup` file uses. The file is provided by EF Portal under `$EF_TOP/<VERSION>/enginframe/plugins/interactive/conf`.

- **basic tools:** bash, grep, cat, printf, gawk, xprop
- **window managers:** metacity, kwin (usually provided by package kdebase), xfwm4
- **dock panels:** tint2, fluxbox, blackbox, mwm (usually provided by package openmotif or lesstif or motif)

Shared file system requirements

Depending on the deployment strategy, EF Portal might require some directories to be shared between the cluster and EF Portal nodes. This guide covers a scenario where both EF Portal Server and EF Portal Agent run on the same host. For more complex configurations or to change the mount points of the shared directories, see the *"Deployment Strategies"* section in the EF Portal Administrator Guide.

In this scenario the EF Portal Server, EF Portal Agent and visualization nodes might require the `$EF_TOP/sessions` directory to be shared. To check if you need to share this directory, see the following table.

Shared File System Requirement

Distributed Resource Manager	Linux®	Windows®
IBM® Platform™ LSF®	Not required	-
SLURM™	Required	-
Altair® PBS Professional®	Required	-

Grid Engine (SGE, SoGE, OGE, UGE)	Required	-

EF Portal Enterprise system requirements

This topic lists the hardware and software prerequisites for an EF Portal Enterprise installation.

Shared file system

The suggested and supported approach to EF Portal Enterprise deployment involves a shared file system for the whole `$EF_TOP` directory tree. Using this approach, you can install and manage the software from just one node, and all the binaries and data directories, such as the spoolers, sessions, and license, are shared among the installation nodes. High-Availability of the Shared File System is consistent with the overall HA/Disaster Recovery strategy.

The NFS `no_root_squash` or equivalent feature must be active to allow the correct management of permissions and ownership of deployed files.

We recommend that you enable on the shared file system the NFS `no_wdelay` or equivalent feature (server-side) to minimize the file writing delay between clients.

Network load balancing

To ensure automated load balancing and high availability for the EF Portal services, it's necessary to set up a network load balancer that dispatches users' requests to the EF Portal Servers in a balanced way.

EF Portal requires the load balancer to implement a *sticky session* strategy. There are many open source and commercial solutions to implement a network load balancer.

For examples of Apache® frontend configurations, see Installing a third-party Load Balancer.

Deployment strategies

Decide how to deploy NI SP EF Portal on your system.

As described in Architectural overview, EF Portal consists of two main software components: the EF Portal Server and the EF Portal Agent.

You can deploy these two components on the same host or on different hosts that communicate across the network. We recommend that you deploy them based on the specifics of your computational resources organization, your network architecture, and your security and performance requirements. In addition, before you deploy them, make sure that the following requirements are met.

- EF Portal Server host must be reachable by HTTP or HTTPS by the clients and the EF Portal Agents.
- EF Portal Agent hosts must have access to your computational resources and your grid infrastructure (for example, to submit jobs to your scheduler).
- EF Portal Server and EF Portal Agent must be installed on a shared storage area.
- For the interactive sessions, EF Portal Server and EF Portal Agent must have read and write access to a storage area shared among them and with the visualization nodes.

If both EF Portal Server and EF Portal Agent run on the same host, communication between the

two is reliable and minimizes administration efforts. We recommend you use this deployment strategy if you have multiple EF Portal installations on different hosts and on each of the hosts you install both an EF Portal Server and Agent. We recommend this deployment strategy for enterprise-level deployments. For more information, see Deployment. Make sure that the EF Portal Servers can communicate with each other.

Depending on your specific use case, you might want to install EF Portal Server and EF Portal Agent on separate hosts. For example, you can run the EF Portal Server in DMZ and EF Portal Agent on the head node of your cluster. For this deployment scenario, make sure that the following conditions are met:

- The Agent and Server must be able to communicate through a TCP connection using the Java™ RMI protocol. The relevant TCP ports that are 9999 for RMI Registry and 9998 for Remote Object, must be free on the Agent's host and reachable from the Server host.
- The Agent and Server must be able to communicate through a TCP connection using the HTTP or the HTTPS protocol. By default, the relevant TCP port is 8080/8443 for HTTP/HTTPS, respectively, on the Server's host must be reachable from the Agent host.
- The Agent and Server must be installed on a shared storage area.
    - The spoolers directory must reside on a storage area to meet the requirements that are described in the "Spoolers Requirements" section of the *EF Portal Administrator Guide*.
    - The sessions directory must reside on a storage area to meet the requirements that are described in EF Portal System requirements.

If you need to access the system through a DMZ, you can use an Apache® web server as frontend in the DMZ when EF Portal is deployed on the intranet. With EF Portal Enterprise, you can additionally use an Apache® server as network load balancer in front of a battery of EF Portal Servers. Make sure that the EF Portal Server is behind the Apache® Web Server that forwards all the EF Portal requests to the Tomcat® servlet container of the EF Portal deployment. For more information, see Apache®-Tomcat® connection.

By default, EF Portal uses Java RMI over SSL between the Server and Agent. You can set up Tomcat® when you install EF Portal to use HTTPS. Alternatively, you can also enable HTTPS after you install EF Portal. For instructions, see Configuring HTTPS.

Installation directories

The *installation directory* is the location, hereafter referred to as `$NISP_ROOT`, where EF Portal binaries, configuration files, and logs are placed.

When you install EF Portal, the following directory structure is created under `$NISP_ROOT`.

```
NISP_ROOT
`-- enginframe
   |-- 2025.0-rXXXX
   | `-- enginframe
   |-- current-version
   |-- bin
   | `-- enginframe
   |-- install
   | `-- 2024.0-rXXXXX
   |-- license
   | `-- license.ef
   |-- conf
   | |-- enginframe.conf
   | |-- enginframe
```

```
| | |-- certs
| | |-- server.conf
| | `-- agent.conf
| |-- tomcat
| | `-- conf
| | `-- certs
| |-- derby
| | |-- derby.properties
| | `-- server.policy
| `-- plugins
|-- data
| |-- cache
| |-- derby
| | `-- efportalDB
| `-- plugins
|-- logs
| `-- <HOSTNAME>
| |-- *.log
| |-- tomcat
| `-- derby
|-- repository
|-- sessions
|-- spoolers
`-- temp
      `-- <HOSTNAME>
      |-- dumps
      |-- errors
      `-- tomcat
```

The following names are used in this guide to refer to the different parts of the EF Portal installation tree.

NISP_ROOT

The directory that contains all the NI SP products. By default, this directory is named `/opt/nisp`.

EF_TOP

The directory that contains the EF Portal product. By default, this directory is named `NISP_ROOT/enginframe`.

EF_LICENSE_PATH

The directory that contains the EF Portal license files. By default, it's named `EF_TOP/license`.

EF_CONF_ROOT

The directory that contains the EF Portal configuration files. By default, it's named `EF_TOP/conf`.

Note
Configuration files can be modified through the Web Configuration Editor.

EF_DATA_ROOT

The directory that contains the data files. By default, it's named `EF_TOP/data`.

EF_LOGS_ROOT

The directory that contains the log files. By default, it's named `EF_TOP/logs`.

Note
Log files can be reviewed using the Log File Viewer service available in the Troubleshooting section of the Operational Dashboard portal.

EF_TEMP_ROOT

The directory that contains the temporary files. By default, it's named `EF_TOP/tmp`.

Note
If you're using or planning to use the AWS HPC Connector plugin, make sure that `EF_TEMP_ROOT` points to a folder on a file system that's shared among the EF Portal nodes.

EF_REPOSITORYDIR

The directory that contains the EF Portal repository files. By default, it's named `EF_TOP/repository`.

EF_SPOOLERDIR

The directory that contains the EF Portal spoolers. By default, it's named `EF_TOP/spoolers`.

INTERACTIVE_SHARED_ROOT

The directory that contains the EF Portal interactive sessions. By default, it's named `EF_TOP/sessions`.

EF_ROOT

The directory that contains the EF Portal binaries and system files. By default, it's named `EF_TOP/<VERSION>/enginframe`.

The PAM based authentication method that's included with EF Portal requires that some binaries have the suid bit set to interact with the underlying system to authenticate users.

If you plan to use this authentication method, make sure that the file system that hosts EF Portal is mounted with `nosuid` flag unset.

The `EF_SPOOLERDIR` directory is used to hold all the data that's supplied as input and created as output by EF Portal services.

As already mentioned, the spooler directory must be accessible by both the EF Portal Server and EF Portal Agent. It must be readable and writable by unprivileged users (described in the following sections) and by the root user.

By default, the spooler directory is placed in a subdirectory of the installation directory.

Managing spoolers contains a detailed description of the System requirements for the spoolers

directory.

Special users

Choose which *system accounts* EF Portal uses.

The EF Portal Administrator is a special system account that has some privileges, such as having access to the EF Portal Operational Dashboard and some of the configuration files. This account is referred to as EF_ADMIN.

Another special account is used to run Tomcat®. This account is referred to as EF_NOBODY. Make sure that all of the configuration files, along with files in the $EF_TOP/logs directory, are owned by EF_NOBODY since they might contain sensitive information. Other users or groups should not be granted permissions to these files.

Any existing system account *excluding root* can be specified. However, we recommend that you set up two new dedicated users for these roles.

In most cases, efadmin and efnobody are respectively used for EF_ADMIN and EF_NOBODY.

EF_ADMIN and EF_NOBODY must be operating system valid accounts. That is, you must be able to log in to the system with those accounts, and they *must not* be disabled.

Authentication

Last, before installing EF Portal, select an authentication method to use.

EF Portal can authenticate users using many different mechanisms, including PAM, LDAP, ActiveDirectory, and HTTP Basic Authentication with certificates.

If the authentication methods included with EF Portal don't meet your requirements, you can create your own authentication module.

For more information about configuring authentication, see Authentication framework.

Distributed Resource Management (DRM) configuration for Interactive Plugin

The following sections describe the additional requirements for the Interactive Portal.

Important
EF Portal periodically checks the status of the interactive jobs using the EF_ADMIN user account. This account must be able to access information from all of the Distributed Resource Management (DRM) mechanisms, such as jobs, files and folder resources.

To make sure that this account can access this information, make sure that this account serves as one of your resource manager administrators and has administrator permissions granted. If you don't configure this account accordingly, you might lose some interactive session data.

Note
When the resource manager controls mixed Windows® clusters, Interactive Plugin submits interactive session jobs on Windows® hosts as the user running EF Portal Server (efnobody by default). This user must be a valid Windows® user.

Note
Starting March 31, 2022, NI SP EF Portal doesn't support VNC®, HP® RGS, VirtualGL, and DCV 2016 and previous versions. Integrations still might be working but are not supported.

IBM® Platform™ LSF®

Interactive Plugin relies on the LSF® workload manager, to allocate and reserve resources to run the interactive sessions. Interactive Plugin requires some specific LSF® settings.

The following is the configuration steps that are necessary to run Interactive Plugin sessions on your LSF® cluster. If needed, they can be enhanced or combined with your existing LSF® configuration to achieve more complex resource sharing policies.

Configuring Queues

Interactive Plugin uses resource manager's queues to submit and manage interactive sessions. You can set up Interactive Plugin to use a default queue and set different services to use different queues. However, it's important that the queues that are used for any visualization middleware or target system have the following settings:

- The EF Portal Administrator account (usually efadmin) must be the queue administrator of any queue used by Interactive Plugin.
- Queues for DCV sessions need to have HJOB_LIMIT set to one, since only one DCV session can run on each host.
- Queues for DCV Linux® sessions need to have PRE_EXEC and POST_EXEC respectively set to the dcv.preexec.sh and dcv.postexec.sh scripts, located under `$EF_TOP/<VERSION>/enginframe/plugins/interactive/tools` folder.

Here is a configuration snippet for these queues in `lsb.queues`. You might not need to have all of these queues configured. You can adapt the parameters as you want, given the preceding requirements.

```
Begin Queue
QUEUE_NAME = dcv_linux
PRIORITY = 50
EXCLUSIVE = y
NEW_JOB_SCHED_DELAY = 0
JOB_ACCEPT_INTERVAL = 0
HJOB_LIMIT = 1
ADMINISTRATORS = efadmin
HOSTS = viz2 vizlin03 vizlin04
PRE_EXEC =
/opt/nisp/enginframe/plugins/interactive/tools/dcv.preexec.sh POST_EXEC
= /opt/nisp/enginframe/plugins/interactive/tools/dcv.postexec.sh
DESCRIPTION = Queue for DCV linux sessions
End Queue
```

Last, the pre-run and post-run scripts for DCV Linux® sessions must run as root. This means that the `/etc/lsf.sudoers` file on all the LSF® nodes must contain the following line: `LSB_PRE_POST_EXEC_USER=root`

Note
Make sure `/etc/lsf.sudoers` is owned by root and has permissions 600. Otherwise, LSF® ignores its contents.
After you modify `/etc/lsf.sudoers`, you must run `badmin hrestart all` to restart sbatchd on all nodes in the cluster.

Note
Edit the $EF_TOP/conf/plugins/ interactive/interactive.efconf file to specify the default dcv queues inside interactive by adding the following two lines.

```
INTERACTIVE_DEFAULT_DCV_LINUX_QUEUE=dcv_linux
INTERACTIVE_DEFAULT_DCV_WINDOWS_QUEUE=dcv_windows
```

Important
By default, every pre-run and post-run script runs with the credentials of the owner of the job. After this configuration is applied, all the pre-execution and post-execution scripts configured in LSF® at queue level (lsb.queues) or at application level (lsb.applications) run with the root account. The impact on security and functionality must be analyzed case by case. Alternatively, it's also possible to configure the **sudo** command to run pre-execution and post execution scripts as a normal user with privileges to run as root only specific operations.

Requirements on scheduler tools

To operate interactive sessions on the target operating systems, EF Portal relies on some tools provided by LSF®. The scheduler must then be properly configured to make these tools work effectively on the hosts of the cluster.

The following is a list of LSF® tools that are required by EF Portal.

- Linux sessions via LSF® require lsrun.

Important
In the recent LSF® versions (specifically, 9.1 and later), the lsrun command is disabled by default. This configuration can be changed by editing file `LSF_TOP/conf/lsf.conf` and setting `LSF_DISABLE_LSRUN=N`. After this change, the LIM daemon must be asked to reload the configuration, as scheduler administrator running the following command: `lsadmin reconfig`

PBS Professional®

Interactive Plugin relies on the PBS Professional® workload manager, to allocate and reserve resources to run the interactive sessions. Installation and configuration instructions for PBS Professional® are out of the scope of this document. However, Interactive Plugin requires some specific PBS Professional® settings.

Here is a minimal configuration needed to run Interactive Plugin sessions on your PBS Professional® cluster. If needed they can be enhanced or combined with your existing PBS Professional® configuration to achieve more complex resource sharing policies.

Configuring Queues

Interactive Plugin uses resource manager's queues to submit and manage interactive sessions. You can set up Interactive Plugin to use a default queue and set different services to use different queues. However, it's important that the queues used for any visualization middleware or target system have the following settings:

- The EF Portal Administrator account (usually efadmin) must be able to see, start, and stop jobs of any queue used by Interactive Plugin.
- You must force the limit of one job per host for DCV queues, because only one DCV session can run on each host.

The following is an example configuration of the interactive queue.

```
# qmgr
Max open servers: 49
Qmgr: create queue interactive
set queue interactive queue_type = Execution
set queue interactive resources_default.arch = linux
set queue interactive enabled = True
set queue interactive started = True
```

SGE, Son of Grid Engine (SoGE), or Univa® Grid Engine® (UGE)

Interactive Plugin relies on the SGE workload manager, to allocate and reserve resources to run the interactive sessions. Installation and configuration instructions for SGE are out of the scope of this document. However, Interactive Plugin requires some specific SGE settings.

Here is a minimal configuration needed to run Interactive Plugin sessions on your SGE cluster. If needed they can be enhanced or combined with your existing SGE configuration to achieve more complex resource sharing policies.

Configuring queues

Interactive Plugin uses resource manager's queues to submit and manage interactive sessions. You can set up Interactive Plugin to use a default queue and set different services to use different queues. However, it's important that the queues that are used for any visualization middleware or target system have the following settings.

- The EF Portal Administrator account (usually `efadmin`) must be the queue administrator of any queue used by Interactive Plugin.
- To make the necessary system command line tools and environment available to Interactive Plugin scripts, SGE queues must be configured as follows:
  - The `shell_start_mode` queue parameter has to be set to `unix_behavior`.
  - If the `shell_start_mode` parameter is set to `posix_compliant`, then the shell parameter must be set to `/bin/bash`.
- You must force the limit of one job per host for DCV queues, because only one DCV session can run on each host.

SLURM™

In this scenario, Interactive Plugin relies on SLURM™ Workload Manager to allocate and reserve resources to run the interactive sessions.

All authorized users must have sufficient permissions to check the status and query all SLURM™ jobs and partitions (alias queues).

Also, Interactive Plugin requires that the `RealMemory` (in MB units) parameter is defined in the SLURM™ configuration for every execution node to show the correct maximum value of memory.

For EF Portal versions prior to 2026.0, the Interactive Plugin requires the feature *dcv2* to be explicitly defined in the SLURM™ configuration. The '*dcv2*' stands for DCV and is no longer mandatory as of EF Portal 2026.0.

Upgrading NI SP EF Portal

EF Portal upgrades are non-disruptive. Running jobs continue without interruption, and active DCV sessions remain fully accessible. All sessions, spoolers, data, logs, and configuration files are preserved throughout the upgrade.

The installer creates parallel installations in `/opt/nisp/enginframe/` or the `$EF_TOP` folder you selected (e.g., `2025.3-r2087`, `2026.0-r2100`).
The active version is controlled by `/opt/nisp/enginframe/current-version` file.

`/opt/nisp/enginframe/install/<old-version>` stores configuration files from previous installations. To maintain consistent paths and settings, it is recommended to provide the previous installation's configuration file to the installer when upgrading to a new version (using the `-f efinstall.config` option).

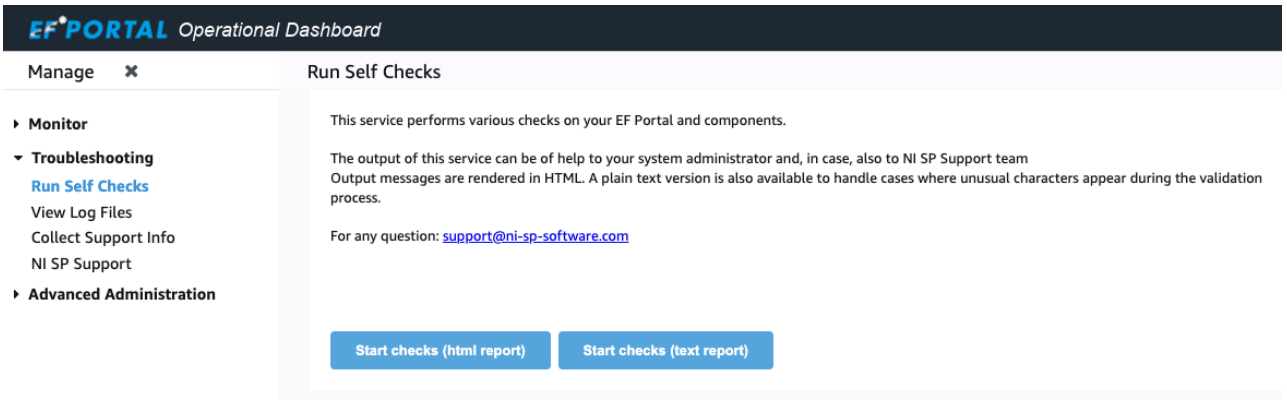
Upgrade Process

Recommended Best Practices:

- Execute *Run Self Checks* service first and review the change report.
- Ensure Java 11+ is installed.
- Back up the license file.
- Prefer "Preserve old version" during installation to allow a controlled switchover.

System Changes Verification

From the *Operational Dashboard*, you can execute the *Run Self Checks* service to review any manual changes applied to system files.



The service performs several checks on the EF Portal installation. In particular, it scans the `EF_ROOT` versioned folder to detect modifications to system files.

The *System files* section highlights:

- **Checksum mismatches**: Files modified in your existing installation (manual migration required).
- **New files**: Custom plugins will be automatically copied to the new installation path, other files can be manually copied at the end of the installation.

If you have any doubts, you can contact NI SP Support.

Prerequisites Verification

Support for Java™ 8 has been discontinued starting with EF Portal 2026.0.

Verify Java version (must be Java 11+):

```
# which java
# java -version
```

System Changes Verification with Dry-Run Installation (optional)

If you cannot execute the *Run Self Checks* service, you can verify the system changes by executing the installer in dry-run mode and stopping it before the actual installation begins. If you have already validated the changes using the previous method, you may skip this step and proceed directly to the installation phase.

Run the installer using the most recent configuration file from the previous installation:

```
# java -jar efportal-<new-version>.jar -f
/opt/nisp/enginframe/install/<previous-version>/efinstall...config
```

Interrupt the process **before the final installation step** by pressing CMD+C or CTRL+C:

```
...
Warning! EF Portal is already installed in /opt/nisp/enginframe/
2025.3-r2087.
Are you sure you want to update? Y or N [default: N]
> Y

============================================================
 EF Portal Update
 Preliminary Checks
============================================================
The installer detected a previous installation in
/opt/nisp/enginframe/2025.3-r2087.
Before updating, the status of the previous installation will be
checked.
This process can take several minutes.
[Press Enter to continue]

Warning! Changes to EF Portal installation in
/opt/nisp/enginframe/2025.3-r2087 have been detected.
A detailed report has been created in: /tmp/enginframe/changes.txt
Please review the changes before continuing.

In case custom plugins have been added to the old EF Portal
installation, the installer will automatically copy them to the new
```

```
installation.
However the installer will NOT apply any additional change from the
old installation files to the new installation.

Are you sure you want to continue the update? Y or N [default: N]
```

Review the generated change report:

```
# cat /tmp/enginframe/changes.txt
```

The `changes.txt` file lists:

- Checksum mismatches: Files modified in your existing installation (manual migration required)
- New files: Custom plugins will be automatically copied to the new installation path, other files can be manually copied at the end of the installation.

Installing the New EF Portal Version

Installer options:

- Auto-switch: Keep both versions in the file system but activate the new version immediately (`current-version` file handled by installer).
- Preserve old version: Install the new version alongside existing but do not switch the version in the `current-version` file (manual switch required later). No service stop required.

Installer behavior:

File Type	Action
Unmodified files	Preserved in the previous version's folder. The new version installs its own copies in a parallel versioned directory, without altering the previous installation
Modified files (checksum mismatch)	Listed in `changes.txt`. Preserved in the previous version's folder. Manual migration required
Custom plugins	Copied automatically
New files	Manual migration required
Sessions/spoolers/data/conf	Preserved (shared directories untouched)
DCV sessions	Unaffected; users continue working
Jobs	Unaffected; continue running

Applying Customizations

When using an external Database, it's required to manually migrate the database connector:

```
# cp -p
```

```
/opt/nisp/enginframe/2025.3-r2087/enginframe/WEBAPP/WEB-INF/lib/mysql
-connector-java-*.jar \
  /opt/nisp/enginframe/2026.0-r2100/enginframe/WEBAPP/WEB-INF/lib/
```

Manually migrate the files listed in `changes.txt`, preserving permissions and ownership. Example: restoring a customized `index.html`:

```
# cp -p /opt/nisp/enginframe/2025.3-r2087/enginframe/index.html \
   /opt/nisp/enginframe/2026.0-r2100/enginframe/index.html
```

Switching the Active EF Portal Version

Required only if you selected "Preserve old version" in Step 2:

1. Stop current version, verifying no processes remain active

```
# /opt/nisp/enginframe/bin/enginframe stop
# ps -f | grep enginframe
```

2. Update version pointer:

```
# echo "EF_VERSION=2026.0-r2100" > \
    /opt/nisp/enginframe/current-version
```

3. Start new version:

```
# /opt/nisp/enginframe/bin/enginframe start
```

Reverting to a Previous Version

1. Stop running version:

```
# /opt/nisp/enginframe/bin/enginframe stop
```

2. Update version pointer back to previous version:

```
# echo "EF_VERSION=2025.3-r2087" > \
     /opt/nisp/enginframe/current-version
```

3. Start previous version:

```
# /opt/nisp/enginframe/bin/enginframe start
```

Directory Structure After Upgrade

Installation Path	Status
/opt/nisp/enginframe/2025.3-r2087/	Previous version (untouched)

/opt/nisp/enginframe/2026.0-r2100/	New version (inactive until switched)
/opt/nisp/enginframe/bin	Shared (untouched)
/opt/nisp/enginframe/current-version	Active version pointer (modified if selected to switch to the new version)
/opt/nisp/enginframe/conf	Shared (existing files untouched)
/opt/nisp/enginframe/data	Shared (existing files untouched)
/opt/nisp/enginframe/license	Shared (license file updated)
/opt/nisp/enginframe/logs	Shared (untouched until switched)
/opt/nisp/enginframe/repository	Shared (untouched until switched)
/opt/nisp/enginframe/sessions	Shared (untouched until switched)
/opt/nisp/enginframe/spoolers	Shared (untouched until switched)
/opt/nisp/enginframe/tmp	Shared (untouched until switched)

Installing NI SP EF Portal

Note
Starting March 31, 2022, NI SP EF Portal doesn't support VNC®, HP® RGS, VirtualGL, and DCV 2016 and previous versions. Integrations still might be working but are not supported.

EF Portal is distributed with an installer that guides you through how to install it. The installer is the EF Portal package itself. For instructions on how to get your EF Portal package, see Downloading EF Portal.

Installing

You can use a graphical and a text-based installer. If you're installing on machines where you don't have access to an X Window System, we recommend that you use a text-based installer.

Unprivileged User Installation
In most cases, we recommend that you install EF Portal as a root user.
Installation as an unprivileged user is possible, but has the following limitations:

- You can't use authentication mechanisms that require root privileges (for example, PAM).
- After the installation, services are run by the user that installed EF Portal Agent.

Make sure that the `umask` is `022` before launching the installation commands.

If Java™ is available in your PATH, start the graphical installer as root user:

```
# java -jar efportal-202*.jar
```

The graphical installer guides you through the installation process. If the X Window System isn't available, the installer falls back to the text-based one.

Start the user interface of the text-based installer by specifying text argument on the command line interface:

```
# java -jar efportal-202*.jar --text
```

The installer shows you the terms of the license agreement. If you aren't installing EF Portal on an Amazon EC2 instance, it also prompts you for a valid license file. If you don't have a valid license file, see EF Portal on on-premises and other cloud-based servers.

Note
The license file is used to determine if the product type of the EF Portal installation is *PRO* or *ENT*. Because EC2 EF Portal doesn't use a license file, you can determine the product type in one of two ways:

- If you're upgrading from an older installation that used a license file, that license is checked. The check determines if the installation is *PRO* or *ENT*. It does this before moving the license file to the backup directory.
- If it's a clean EF Portal installation, then you're prompted to choose between a *PRO* and *ENT* installation.

In either case, a field that's called *ef.product* with the value of *PRO* or *ENT* is written in the `$EF_TOP/conf/enginframe/server.conf` config file.

The installer prompts you with some questions to tailor the EF Portal deployment to your needs.

After asking all the questions, the installer shows you a summary of your answers. This is when you can change values before installing. After the summary is accepted, the installer proceeds to set up the EF Portal on the current host.

Note
When installing using a Java™ Runtime Environment the following warning might appear in the output: `Unable to locate tools.jar. Expected to find it in [...]` This warning is harmless and can be safely ignored.

After installation finishes, you can begin to use EF Portal. To learn how you can get started with your EF Portal Portal and test your installation, see Running NI SP EF Portal.

A file that's named `efinstall.config` is saved in the directory where you launched the installer from. This file contains the options that you specified during installation. This file can be useful to document how you installed EF Portal. You can use this information to replicate installation in *batch* mode without requiring user interactions.

Distributed Deployment
If you want to run Server and Agent on different hosts, launch the installer on the Server host and select a shared installation directory. The installer asks if the Server and Agent run on different hosts.

Batch Installation

When using batch installation, you can replicate an installation on different hosts taking as input a configuration file that's created during a normal install. To run the EF Portal installer in batch mode, run the following command.

```
# java -jar efportal-202*.jar --batch -f efinstall.config
```

If you didn't specify the `-f` option, the installer searches for a file that's named `efinstall.config` in the current directory.

Unless errors occur, EF Portal installer completes the installation procedure without requiring further user input.

Fine-tuning your installation

We recommend that you consider fine-tuning your installation. You can make adjustments to make sure that your installation of EF Portal is performing optimally on your system and meets your specific requirements.

Topics

Spooler download URL

The EF Portal Agent must communicate with EF Portal Server when downloading a file from a spooler. In some network configurations and architectures, when a Web Server is placed in front of the EF Portal Server, this callback URL must be explicitly configured.

If you can't download files from your EF Portal Portal, see Configure download URL on agent for configuring EF Portal Agent callback URL.

Optimizing JDK options

To fine tune the JVM heap size used to launch the EF Portal Server or the EF Portal Agent, you can modify settings in the `$EF_TOP/conf/enginframe.conf` file. You can set `SERVER_MIN_HEAP` and `SERVER_MAX_HEAP` for the EF Portal Server, and `AGENT_MIN_HEAP` and `AGENT_MAX_HEAP` for the EF Portal Agent.

If you want full control of the JVM options, you can set `SERVER_JAVA_OPTIONS` and `AGENT_JAVA_OPTIONS` in the `$EF_TOP/conf/enginframe.conf` file. However, note that doing so overrides *all* of the parameters, including the default ones.

In the following example, the heap size for Agent is set to 1024 MB.

```
# Initial heap memory size for the EF Portal Agent
AGENT_MIN_HEAP='1024m'

# Max heap memory size for the EF Portal Agent
AGENT_MAX_HEAP='1024m'
```

The EF Portal Installer follows a consumption safe logic to calculate optimal heap memory limits, which is the following:

- available memory on the host (in MB) is grabbed looking at `MemAvailable` in `/proc/meminfo` (so not the global amount of memory physically installed on the machine)
- it is then divided by a fixed int factor to avoid filling the whole memory
    - factor is currently **3**
- we search in a memory setting list to get the nearest entry which matches the available memory
- anyway we cannot go below or above the threshold defined in the setting list itself.

Current thresholds are:

- min: Server 1GB, Agent 512MB

- max: Server 3GB, Agent 1GB

Below is the memory setting list:

Server	Agent
1024	512
1536	512
2048	768
2560	768
3072	1024

If computed settings are greater than current ones already defined in `enginframe.conf` they are updated in place.

Distributed resource manager options

Enabling Grid Managers

Before using any HPC or VDI services in EF Portal, you must enable the Grid Manager through the Web Configuration editor (introduced in EF Portal 2026.0) or by editing the grid manager configuration file (`$EF_TOP/conf/plugins/grid/grid.conf`).

EF_GRID_MANAGERS — Comma-separated list of HPC workload managers and VDI session managers available in EF Portal.

Supported values:

- `slurm` — SLURM (HPC and VDI for Linux)
- `lsf` — IBM LSF (HPC and VDI for Linux)
- `pbs` — Altair PBS Professional (HPC and VDI for Linux)
- `sge` — Grid Engine (HPC and VDI for Linux)
- `torque` — TORQUE (HPC and VDI for Linux)
- `dcvsm` — DCV Session Manager (VDI for Windows and Linux)

Example: `EF_GRID_MANAGERS="slurm,dcvsm,lsf"`

EF_DEFAULT_GRID_MANAGER — Default scheduler/workload manager used when no specific grid manager is specified during job submission. This value must match one of the managers listed in EF_GRID_MANAGERS. If unset or empty, EF Portal automatically uses the first entry from the EF_GRID_MANAGERS list.
Example: `EF_DEFAULT_GRID_MANAGER="slurm"`

Prerequisites

1. Enable at least one grid manager in `EF_GRID_MANAGERS` before using HPC or VDI services
2. Configure the selected grid manager using its specific plugin configuration (see the individual workload manager sections below)

3. Set `EF_DEFAULT_GRID_MANAGER` to ensure predictable behavior during job submissions
4. Without these settings, HPC and VDI services will not function in the EF Portal.

SLURM Plugin configuration

All the following SLURM configuration parameters can also be managed through the Web Configuration editor introduced in EF Portal 2026.0.

The SLURM plugin is configured in its main configuration file, located under the EF Portal plugin configuration directory (i.e., `$EF_TOP/conf/plugins/slurm/ef.slurm.conf`). You must configure these parameters before using SLURM-based services in your EF Portal environment.

SLURM_BINDIR — Absolute path to the directory containing the SLURM client binaries (for example, squeue, sacct, sbatch). If not set, EF Portal relies on the system PATH to find these commands.

SLURM_CLUSTER_LABEL — Human-readable label for the SLURM cluster that's shown in the EF Portal UI. If unset, EF Portal uses the cluster name returned by SLURM.

SLURM_INTERACTIVE_FEATURE — SLURM feature constraint applied to interactive session jobs. When you set this parameter, its value is passed as `--constraint` at submission time so that sessions run only on nodes matching the specified feature (for example, dcv2 for DCV-capable nodes).

SLURM_INTERACTIVE_SHARED_ROOT_EXEC_HOST — Path to the shared directory as it appears on the execution hosts. Use this when the interactive shared root is mounted at a different location on compute nodes than on the EF Agent host, so that interactive jobs can correctly access shared data.

SLURM_SUBMISSION_DIR — Directory used as the working directory when `grid.submit` sends jobs to SLURM. If this parameter is not set, EF Portal uses the current working directory at submission time (typically the spooler directory).
On systems where the spoolers are not shared with execution hosts, you must set this to a path that's visible from both the submission host and the execution nodes (for example, a shared /tmp or project directory).

SLURM_JOB_FORMAT — Multiline format string that defines the fields returned by the squeue command. Each line contains one SLURM %-style field specifier (such as `JobId=%i` or `WorkDir=%Z`). EF Portal parses this output using an AWK helper script to produce GridML job descriptions, so if you add, remove, or reorder fields in this format you must update the corresponding AWK script to keep the field-to-XML mapping consistent.

SLURM_SACCT_JOB_FORMAT — Comma-separated list of sacct field identifiers (with optional format modifiers like `JobName%30s`) used when EF Portal queries job accounting data. Fields in this list are strictly positional. Changing the order, adding, or removing fields requires you to adjust any parsing logic that depends on the original order.

LSF Plugin configuration

All the following LSF configuration parameters can also be managed through the Web Configuration editor introduced in EF Portal 2026.0.

The LSF plugin is configured in its main configuration file, located under the EF Portal plugin configuration directory (i.e., `$EF_TOP/conf/plugins/lsf/ef.lsf.conf`). You must configure these parameters before using LSF-based services in your EF Portal environment.

LSF_PROFILE — Absolute path to the profile script that sets the environment required to run LSF commands. Leave this parameter empty on Windows environments. When set, EF Portal sources this file so that LSF client tools are available with the correct environment.

LSF_WIN_BIN — Absolute path to the directory that contains LSF binaries on Windows (for example, `C:\LSF_7.0\7.0\bin`). Leave this parameter empty on non-Windows environments. EF Portal uses this path to locate LSF commands when running on Windows.

LSF_LOAD_PROFILE — Controls when the LSF profile is loaded. Valid values are:

- `login` (load the LSF profile at user login, default),
- `always` (load the profile at every service execution, matching older EF Portal behavior),
- `false` (do not load the profile because the user shell already does it). Choose `` `false` `` if your login environment is already LSF-aware.

USE_PROJECT_FOR_USER — When set to `true`, EF Portal assigns `` `-P $EF_USER` `` to each job submitted through the LSF plugin and only operates on jobs whose LSF project equals the EF user name. Set this to `true` when you want per-user project isolation for job operations; the default is `false`.

EF_KILL_JOB_ON_SPOOLER_REMOVAL — When set to `true`, EF Portal automatically kills all jobs that were submitted via the LSF plugin when their associated spooler is removed. Use this if you want the spooler lifecycle and LSF job lifecycle to be tightly coupled; otherwise, leave it `false`.

LSF_STRIP_DOMAIN — Space-separated list of domain suffixes (each starting with a dot) that EF Portal strips from hostnames. Use this when hosts have fully qualified domain names across one or more domains and you prefer to display or match short hostnames. For example: `` `.ni-sp.com .ni-sp-software.com` ``. Leave it empty to keep hostnames unchanged.

LSF_CLUSTER_LABEL — Human-readable label for the LSF cluster that EF Portal shows in the UI. If you don’t set this parameter, EF Portal uses the cluster name reported by LSF. Set a custom label when you want a friendlier or more descriptive cluster name.

LSF_MULTICLUSTER — Enables the LSF multicluster view in cluster load services. When set to `true`, EF Portal queries and displays load information from multiple LSF clusters in a unified view. Use this in environments where LSF multicluster is configured.

LSF_MULTICLUSTER_THR_MASTER — When set to `true`, `lsrun` requests to remote clusters are routed through the local master host. This is required in some LSF multicluster setups where direct cross-cluster connectivity is restricted or unsupported. Leave it unset or `false` if your environment doesn’t need this behavior.

LSF_INTERACTIVE_USE_SHARED_FS — Forces interactive jobs to use a shared filesystem instead of `bread/bpost` for data exchange between execution hosts and EF Portal. By default, standard LSF uses `bread/bpost`, while OpenLava relies on a shared filesystem. Set this to `true` if you want to standardize on a shared filesystem for all interactive jobs.

LSF_BPOST_MIN_INDEX — Base index used when EF Portal attaches `bpost` messages to jobs. EF Portal will use this index and higher values when sending job-related messages; adjust it only if you need to avoid collisions with other tools or conventions that also use `bpost` indexes.

LSF_INTERACTIVE_SHARED_ROOT_EXEC_HOST — Path to the shared directory as seen from the execution hosts. Use this parameter when the interactive shared root is mounted in a different location on compute nodes than on the EF Agent host, so that interactive jobs can correctly access shared data on the cluster nodes.

LSF_INTERACTIVE_RESOURCE — LSF resource string applied to interactive session requests. When set, EF Portal passes this value as a resource requirement at submission time (for example, to target GPU nodes or specific host groups). Leave it empty if you don’t need additional resource constraints.

LSF_INTERACTIVE_DO_NOT_QUEUE — When set to `true`, EF Portal checks for available candidate hosts before submitting an interactive job and fails immediately if none are available instead of queuing the request. Enable this behavior when you want interactive sessions to start only if capacity is available right away, and prefer a fast failure over queueing.

PBS Plugin configuration

All the following PBS configuration parameters can also be managed through the Web Configuration editor introduced in EF Portal 2026.0.

The PBS plugin is configured in its main configuration file, located under the EF Portal plugin configuration directory (i.e., `$EF_TOP/conf/plugins/pbs/ef.pbs.conf`). You must configure these parameters before using PBS-based services in your EF Portal environment.

PBS_BINDIR — Absolute path to the directory that contains the PBS client binaries (for example, qsub, qstat, tracejob). When set, EF Portal uses this location to run PBS commands instead of relying solely on the system PATH.

PBS_CLUSTER_LABEL — Human-readable label for the PBS cluster that EF Portal displays in the UI. If this parameter isn’t set, EF Portal uses the cluster name reported by PBS. Configure this when you want a more descriptive or user-friendly cluster name.

PBS_SHOW_ALL_FINISHED — Controls whether finished jobs are shown in the “all jobs” view. This parameter interacts with the EF Portal job cache: when set to true, EF Portal attempts to expose finished jobs as long as they are still present in the job cache or underlying scheduler history. Set it to false if you want a more compact view focused on active and recently finished jobs only.

PBS_INTERACTIVE_SHARED_ROOT_EXEC_HOST — Path to the shared directory as seen from the execution hosts. Use this when the interactive shared root is mounted at a different

location on compute nodes than on the EF Agent host, so that interactive sessions can correctly access shared data on those nodes.

PBS_INTERACTIVE_LINUX_ARCH — Maps the interactive “Linux” OS option in EF Portal to the appropriate PBS resource string (for example, `arch=linux`). EF Portal adds this resource to interactive job submissions to ensure that sessions are scheduled on Linux nodes. Adjust it to align with the resource naming in your PBS environment.

PBS_SUBMISSION_DIR — Directory used as the working directory when grid.submit sends jobs to PBS. If not set, EF Portal uses the current working directory at submission time (typically the EF spooler directory). In environments where EF Portal spoolers are not shared with execution hosts, job submissions may fail unless you set this to a path that’s available on both the submission host and all execution nodes (for example, a shared /tmp or project directory).

PBS_JOB_HISTORY_INTERVAL — Number of days in the past that job history is queried using `tracejob -n <days>`. The default in tracejob is 1 day; EF Portal’s default value here is 14 days. Increase this value if you need to inspect older jobs through EF Portal, being aware that larger history windows can increase the time required to run history queries.

SGE Plugin configuration

All the following SGE configuration parameters can also be managed through the Web Configuration editor introduced in EF Portal 2026.0.

The SGE plugin is configured in its main configuration file, located under the EF Portal plugin configuration directory (i.e., `$EF_TOP/conf/plugins/sge/ef.sge.conf`). You must configure these parameters before using SGE-based services in your EF Portal environment.

SGE_PROFILE — Absolute path to the profile script that sets up the environment required to run SGE commands (for example, `/opt/sge/default/common/settings.sh`). When set, EF Portal sources this file so that SGE client tools are available with the correct environment variables and PATH.

EF_SGE_JOB_LIST_MODE — Controls how SGE jobs are listed and whether finished jobs are displayed. Supported values:

- `default`: Shows finished jobs using qstat + qacct (may be slow on clusters with many jobs due to qacct queries).
- `qacct-on-my-jobs`: Runs qacct only for the current user’s jobs or job/host detail pages (recommended for moderate-sized clusters).
- `no-qacct`: Never runs qacct and does not show finished jobs (fastest option).
- `brief`: Shows limited job details without finished jobs (recommended for clusters with many jobs).

SGE_CLUSTER_LABEL — Human-readable label for the SGE cluster shown in the EF Portal UI. If not set, EF Portal uses the cluster name reported by SGE. Use this parameter when you want a friendlier or more descriptive cluster name.

QACCT_DAYS — Number of days in the past for which `qacct` queries accounting data (`-d days` option). This value is passed to every qacct invocation by the plugin. Increase it if you need historical accounting data beyond the default of 1 day.

QACCT_CUSTOM_PARAMS — Additional custom parameters passed to every qacct invocation by the plugin. Do not include the `-d days` option here, as it's already handled by QACCT_DAYS. Use this for cluster-specific qacct flags or filters.

SGE_INTERACTIVE_SHARED_ROOT_EXEC_HOST — Path to the shared directory as seen from the execution hosts. Use this parameter when the interactive shared root is mounted at a different location on compute nodes than on the EF Agent host, so that interactive sessions can correctly access shared data on those nodes.

TORQUE Plugin configuration

All the following Torque configuration parameters can also be managed through the Web Configuration editor introduced in EF Portal 2026.0.

The Torque plugin is configured in its main configuration file, located under the EF Portal plugin configuration directory (i.e., `$EF_TOP/conf/plugins/torque/ef.torque.conf`). You must configure these parameters before using Torque-based services in your EF Portal environment.

PBS_BINDIR — Absolute path to the directory that contains the Torque client binaries (for example, qsub, qstat, tracejob). EF Portal uses this path to locate Torque commands instead of relying solely on the system PATH.

TORQUE_CLUSTER_LABEL — Human-readable label for the Torque cluster shown in the EF Portal UI. If you don't set this parameter, EF Portal uses the cluster name reported by Torque. Configure this when you want a friendlier or more descriptive cluster name.

TORQUE_INTERACTIVE_SHARED_ROOT_EXEC_HOST — Path to the shared directory as seen from the execution hosts. Use this parameter when the interactive shared root is mounted in a different location on compute nodes than on the EF Agent host, so interactive sessions can correctly access shared data on the cluster nodes.

TORQUE_INTERACTIVE_LINUX_ARCH — Maps the interactive "Linux" OS option in EF Portal to the appropriate Torque resource string (for example, `arch=linux`). This resource is added to interactive job submissions so that sessions land on Linux nodes. Adjust it to match the resource naming in your Torque environment.

TORQUE_JOB_HISTORY_INTERVAL — Number of days in the past that Torque job history is queried using `tracejob -n <days>`. The default in Torque is 1 day; EF Portal's default here is 14 days. Increase this value if you need to inspect older job history through EF Portal, bearing in mind that larger windows may increase query times.

HPC Connector configuration

The EF Portal installer configures the HPC Connector at installation time. You can modify the HPC Connector parameters in the AWS HPC Connector (hpc) main configuration file. It's located at `$EF_TOP/conf/plugins/hpc/hpc.efconf`. This is required as a preconfiguration step before the HPC Connector can be used within your EF Portal environment.

HPCC_AWS_PROFILE_NAME — The AWS profile name that's used for creating the clusters.

HPCC_AWS_REGION — The AWS Region where the clusters will be created by default.

HPCC_AWS_BUCKET_ARN — The Amazon Resource Name (ARN) of the Amazon S3 bucket that's used for data exchange.

HPCC_SSM_ROLE_ARN — The ARN of the IAM role that's used for invoking remote commands using SSM.

HPCC_S3_ROLE_ARN — The ARN of the IAM role that's used for data exchange.

HPCC_PCLUSTER_ROLE_ARN — The ARN of the IAM role that's used for creating the clusters using AWS ParallelCluster.

HPCC_USE_INSTANCE_PROFILE — Set this boolean parameter to true if EF Portal is installed on an Amazon EC2 instance and you want the HPC Connector to use an instance profile instead of the profile mapped to the ef-iam-user. The default value for this parameter is false. For information about how to set up an instance profile, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the AWS Identity and Access Management User Guide.

HPCC_DELETED_CLUSTER_TTL the time to live after which a cluster in the deleted state will be removed from the system. The time span must follow the ISO8601 duration syntax. The default value for this parameter is one day.

HPCC_COMPLETED_APPLICATION_TTL the time to live after which a completed remote job is removed from the system. The time span will need to follow the ISO8601 duration syntax. The default value for this parameter is seven days.

Cluster name label

All the grid plug-ins in EF Portal support the option to specify a custom label to be used by the portal to show the name of the clusters.

Change the grid plug-in cluster name by specifying in the plug-in configuration file the `[PLUGIN ID]_CLUSTER_LABEL` options:

- `$EF_TOP/conf/plugins/lsf/ef.lsf.conf: LSF_CLUSTER_LABEL=...`
- `$EF_TOP/conf/plugins/pbs/ef.pbs.conf: PBS_CLUSTER_LABEL=...`
- `$EF_TOP/conf/plugins/sge/ef.sge.conf: SGE_CLUSTER_LABEL=...`
- `$EF_TOP/conf/enginframe/plugins/slurm/ef.slurm.conf: SLURM_CLUSTER_LABEL=...`

The label can be managed through the Web Configuration editor introduced in EF Portal 2026.0.

Configuring multiple clusters for the same grid manager

Currently, EF Portal supports the multi-cluster configuration only for the Slurm plug-in.

Configure multiple Slurm clusters by specifying following options in the plug-in configuration file `$EF_TOP/conf/enginframe/plugins/slurm/ef.slurm.conf`:

SLURM_CLUSTER_IDS

Comma-separated list of IDs that are assigned to a cluster.

SLURM_CLUSTER_[CLUSTER-ID]_LABEL

(Optional) Label to display for the cluster that's identified by the CLUSTER-ID. CLUSTER-ID is used if the label isn't set.

SLURM_CLUSTER_[CLUSTER-ID]_CONF

For each cluster, the path to the configuration file of the cluster is identified by CLUSTER-ID.

The following is an example.

```
# Optional configuration: if not set, only the default cluster will be
used, with following settings:
#SLURM_CLUSTER_LABEL
# (optional, cluster id will be used if not set)

SLURM_CLUSTER_IDS="SLURM1,SLURM2"

# First Cluster
SLURM_CLUSTER_SLURM1_LABEL="SLURM_FIRST"
SLURM_CLUSTER_SLURM1_CONF="/efs/slurm/slurm_1/etc/slurm.conf"

# Second Cluster
SLURM_CLUSTER_SLURM2_LABEL="SLURM_SECOND"
SLURM_CLUSTER_SLURM2_CONF="/efs/slurm/slurm_2/etc/slurm.conf"
```

Important
You can't specify the `SLURM_CLUSTER_LABEL` option if multiple Slurm clusters are used. You can only use clusters that are compatible with the Slurm client in the folder indicated by the `SLURM_BINDIR` option.

Interactive plugin

Distributed resource manager

The EF Portal installer configures the Interactive Plugin depending on the resource manager selection when you install EF Portal. If you want to change this setting, change the configuration parameters located in the `$EF_TOP/conf/plugins/interactive/interactive.efconf` file.

The parameters are `INTERACTIVE_DEFAULT_LINUX_JOBMANAGER` for Linux and `INTERACTIVE_DEFAULT_WINDOWS_JOBMANAGER` for Windows. You can set their values to the following:

- `lsf` - Sessions are scheduled by LSF® (*Linux®-only*).
- `pbs` - Sessions are scheduled by PBS Professional® (*Linux®-only*).
- `sge` - Sessions are scheduled by Sun® Grid Engine, Univa® Grid Engine® (UGE) or Son of Grid Engine (SoGE) (*Linux®-only*).

You can override this behavior on a per-EF Portal service basis by using `--jobmanager` option of interactive.submit session starter script.

Remote visualization technology

By default, Interactive Plugin is set to use DCV remote visualization technology. If you want to change this setting, the related configuration parameter is located in the Interactive Plugin main

configuration file: `$EF_TOP/conf/plugins/interactive/interactive.efconf`.

The parameter is named `INTERACTIVE_DEFAULT_REMOTE`. You can set it to `dcv2` to manage DCV 3D accelerated sessions.

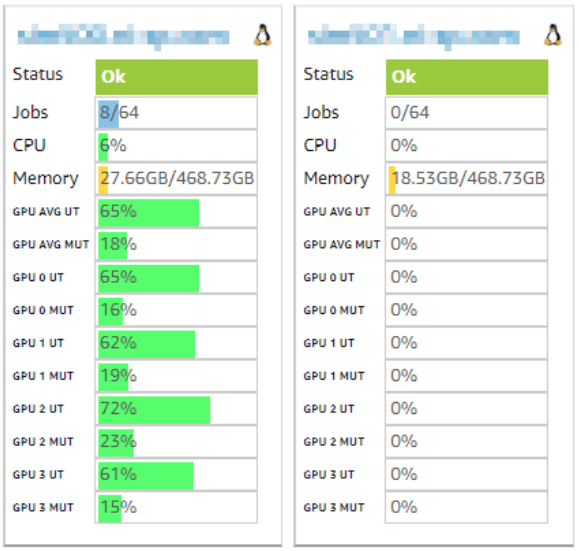
You can override this behavior on a per-EF Portal service basis by using the `--remote` option of the `interactive.submit` session starter script.

GPU Monitoring

EF Portal supports monitoring GPU compute load and memory load. Average of the GPU compute load and memory load over all GPUs is provided and visualized as well as the compute load and memory load of the individual GPUs.

GPU monitoring is supported for LSF with native GPU integration and custom monitoring for all other schedulers like SLURM, DCV Session Manager and others.

Here is a sample screenshot in the hosts list view:



For other schedulers than LSF without GPU load integration the GPU load can be collected and stored for visualization in the EF Portal.

EF Portal will automatically display GPU load data stored in the directory `$EF_TEMP_ROOT/gpumon` (e.g. `/opt/nisp/enginframe/tmp/gpumon`).

The file format of the files is `gpu-log.$HOSTNAME` where the hostname should match the hostname displayed in the EF Portal host list.

The data can be collected on the individual GPU servers e.g. with the following command which can be run in crontab (please replace `EF_TEMP_ROOT` with the respective value in your environment and create the directory `$EF_TEMP_ROOT/gpumon` with the right permissions so that the crontab processes can write the files):

```
nvidia-smi
--query-gpu=index,utilization.gpu,utilization.memory,temperature.gpu
--format=csv,noheader,nounits                                        >>
```

```
$EF_TEMP_ROOT/gpumon/gpu-log.$HOSTNAME
0, 10, 12, 34
```

For DCV Session Manager GPU load integration a data collection script is available on request from NI SP.

Installing EF Portal Enterprise

Topics

Installing a third-party Load Balancer

EF Portal Enterprise requires a load balancer implementing the sticky session strategy. The following sections describe how you can implement load balancing for EF Portal Enterprise. This implementation is based on the AJP connector and Apache® Web Server frontend with `mod_proxy_balancer` module.

Topics

Configure the AJP connector

Enable the AJP connector on Tomcat®:

- Login as root on the node of the EF Portal server:

```
# cd $EF_TOP/conf/tomcat/conf
```

- Open the `server.xml` file and add a section to define the AJP 1.3 connector.

```
<Connector
 port="8009"
 secretRequired="false"
 enableLookups="false"
 redirectPort="8443"
 protocol="AJP/1.3"
 URIEncoding="utf-8" />
```

If port 8009 is used by another application, choose another port.

Configure the Apache® Proxy

To enable Reverse Proxy Support in Apache® append the following lines to the main Apache® configuration file (`APACHE_TOP/conf/httpd.conf`) and reload the Apache® service.

```
<Location "/enginframe">
 DefaultType None
 ProxyPass ajp://127.0.0.1:8009/enginframe flushpackets=on
 ProxyPassReverse ajp://127.0.0.1:8009/enginframe
</Location>
```

If your context isn't `enginframe`, then change the `<Location>` and those two lines accordingly.

Configure the Apache® mod_proxy_balancer

This configuration is required to balance the traffic over many EF Portal instances.

Create a specific file (for example, `ef-ent.conf`), and add to the Apache® configuration directory. In most cases, it's `/etc/httpd/conf.d`.

The following is an example of the `ef-ent.conf` file content.

```
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e;
path=/efportal" env=BALANCER_ROUTE_CHANGED

<Proxy balancer://enterprise>
 BalancerMember ajp://<ip of EF Portal Server 1>:8009 route=1
BalancerMember ajp://<ip of EF Portal Server 2>:8009 route=2
 ProxySet lbmethod=bybusyness
 ProxySet stickysession=ROUTEID
</Proxy>

<Location "/enginframe">
 ProxyPass balancer://<enterprise hostname>/enginframe
 ProxyPassReverse balancer://<enterprise hostname>/enginframe
</Location>
```

The enterprise alias is an internal parameter. If necessary, you can change the efportal context. You must set up the Header to internally store the route ID to pass it to stickysession variable, useful for automatic management of cookies.

Setting up the database management system

An empty database instance that's named `EnginFrameDB` must be created before EF Portal first startup. `EnginFrameDB` is case sensitive. At the first startup, EF Portal creates all the necessary tables.

The following sections describe the steps to create the `EnginFrameDB` database instance on the supported database management system.

Note
Don't start an EF Portal server or servers before the following steps are completed.

Topics

Install and configure EF Portal

When installing EF Portal Enterprise, you're asked to insert the JDBC URL to the EF Portal database instance. You're also asked to add the username and password that you use to access it.

```
JDBC URL [default: jdbc:derby://localhost:1527/EnginFrameDB]
> jdbc:mysql://172.16.10.216:3306/EnginFrameDB
Username [default: dbadmin]
> efportaldb
Password
>
```

Configure the MySQL® database (version 5.1.x and higher)

1. Use the following command on the MySQL® server host to log in as root.

```
# mysql -p
```

2. Create a new database by running the following SQL query for MySQL 5.x.

```
# CREATE DATABASE EnginFrameDB;
```

   For MySQL 8.x and later, you must define a character set that isn't UTF8 due to a limitation on row length.

   Installing EF Portal Enterprise

```
# CREATE DATABASE EnginFrameDB DEFAULT CHARACTER SET latin1;
```

3. Create a new user by running the following SQL queries, using single quotes.

```
# CREATE USER '<username>'@'<host>' IDENTIFIED BY '<password>';
```

   Consider the following example.

```
# CREATE USER 'efdbadmin' IDENTIFIED BY 'efdbpassword';
# CREATE USER 'efdbadmin'@'%' IDENTIFIED BY 'efdbpassword';
```

   The first statement allows access to localhost (for example, for maintenance actions). In the second line, '%' functions as a wildcard that's used to allow all hosts. You can modify the latest statement to restrict the access to the EF Portal Servers only.

4. Grant privileges to the new user on the previously created DB by running the following SQL query.

```
# GRANT ALL PRIVILEGES ON EnginFrameDB.* TO <username>
 IDENTIFIED BY '<password>';
```

   Consider the following example.

```
# GRANT ALL PRIVILEGES ON EnginFrameDB.* TO 'efdbadmin'
 IDENTIFIED BY 'efdbpassword';
# GRANT ALL PRIVILEGES ON EnginFrameDB.* TO 'efdbadmin'@'%'
 IDENTIFIED BY 'efdbpassword';
```

5. Flush privileges to activate them on the created database:

```
# flush privileges;
```

6. Test the connection to the `EnginFrameDB` database. From one of the servers where the EF Portal server instance is installed on, check the connection to the created database using MySQL® client only.

```
# mysql -h <mysql server hostname/ip address>[:<port>]
 -u <username>
 -p <password>
```

Consider the following example.

```
# mysql -h mysqlserver -u efdbadmin -p efpassword
```

This message is returned.

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 87653
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql>
```

Configure the Oracle database (10 and higher)

- Log in to the database as admin using a SQL client such as the SQuirreL SQL client, using the URI `jdbc:oracle:thin:@<hostname>:<port>:XE` (for example, `jdbc:oracle:thin:@efentserverdb:1521:XE`).
- Create a new user by running the following SQL query.

```
# CREATE USER <username> IDENTIFIED BY <password>
```

- The following is an example.

```
# CREATE USER efportaldb IDENTIFIED BY efdbpassword
```

A new schema with name equals to the username is automatically created by the db engine. Logging into the database with the new user, the user schema is automatically used for that user.

- Grant privileges to the new user by running the following SQL query.

```
# GRANT ALL PRIVILEGES TO <username>
```

- The following is an example.

```
# GRANT ALL PRIVILEGES TO efportaldb
```

If you need to restrict some privileges for the new user, see the instructions in the Oracle database documentation. Make sure that the user has both read and write permissions on their schema.

- Change the database user profile to avoid automatic expiration, issuing the following SQL query.

```
# ALTER PROFILE DEFAULT LIMIT PASSWORD_LIFE_TIME UNLIMITED;
```

This sets the *no password expiration* for all the default user's profile.

Configure the Microsoft SQL Server® (2012, 2008)

Follow these steps to configure Microsoft SQL Server for EF Portal.
SQL Server might require additional database settings to prevent deadlock issues with EF Portal's job cache operations.

Step 1: Basic SQL Server Setup

- Enable TCP/IP networking and remote connections using SQL Server Configuration Manager:
  - Go to SQL Server Network Configuration > Protocols for SQLEXPRESS
  - Enable TCP/IP
  - Open the context menu (Right-click) on TCP/IP and select Properties
  - Verify that, under IP2, the IP address is set to the computer's IP address on the local subnet
  - Scroll down to IPAll
  - Make sure that TCP Dynamic Ports is blank
  - Make sure that TCP Port is set to 1433
- Create a new database named `EnginFrameDB` using the SQL Server Management Studio with default settings.
- Create a user with SQL Server® authentication using the SQL Server Management Studio (e.g. `efdb` with password `efdbpassword`).
  - Assign `db_owner` role on the EnginFrameDB. Otherwise, it can't read and create tables. Check this in the Object Explorer. Go to Security > Logins, open a content menu (right-click) on the username and choose Properties > User Mapping from the left panel.
  - Make sure that both the EnginFrameDB database option and the role membership option are checked (for example, have both the db_owner and public options checked).
  - (optional) To set EnginFrameDB as the default database for the user, go to Security > Logins, open a context menu (right-click) on the username and choose Properties > General from the left panel.
- Allow SQL Server® to be accessed through "SQL Server and Windows Authentication mode" and not only through "Windows authentication mode"
  - Open a context menu (right-click) on the database server instance where EnginFrameDB was created (for example, SQLEXPRESS).
  - Choose Properties, then Security, Server authentication and choose "SQL Server and Windows Authentication mode".

At this point, EF Portal must be able to access SQL Server® databases and to create tables.

Step 2: Required Anti-Deadlock Configuration

SQL Server's default READ COMMITTED isolation blocks readers and writers, causing deadlocks in EF Portal's high-concurrency job cache operations (e.g., "*Transaction was deadlocked... deadlock victim*").

Enable snapshot isolation to use row versioning instead of locks:

```
# ALTER DATABASE EnginFrameDB SET ALLOW_SNAPSHOT_ISOLATION ON;
# ALTER DATABASE EnginFrameDB SET READ_COMMITTED_SNAPSHOT ON;
```

ALLOW_SNAPSHOT_ISOLATION ON — Enables the snapshot isolation mechanism at the database level, activating row versioning in tempdb.
READ_COMMITTED_SNAPSHOT ON — Changes the default READ COMMITTED isolation level to use row versioning instead of shared locks for SELECT statements.

Configure the EF Portal service

The installer can configure the EF Portal service to start at boot time on the node where it's run.

If the installation was performed on a shared file system, log in on each node and run the following command to configure the EF Portal service to start on all other dedicated nodes.

```
# $EF_TOP/bin/enginframe host-setup --roles=server,agent --boot=y
```

This configuration starts both server and agent components at node startup.

To learn about the host-setup options, run the following command.

```
# $EF_TOP/bin/enginframe host-setup --help
```

Start EF Portal

After the EF Portal service is installed, run the following command to manually start it on every node.

```
# service enginframe start
```

or

```
# systemctl start enginframe
```

Check if every database EnginFrameDB instance (MySQL®, Oracle® or SQL Server®) has been populated with EF Portal tables.

Saving database credentials in a keystore

By default, when you're configuring an external database, the database credentials are written to the `server.conf` configuration file. The option names `ef.db.admin.name` and `ef.db.admin.password` are set to the database username and password. You can save the

database credentials in a keystore instead of the configuration file.

Edit the `${EF_ROOT}/enginframe/server.conf` file to set up the keystore option. The resulting file includes the following contents.

Configure EF Portal to work behind a network proxy

```
EF_DB_KEY_STORE_ENABLED=true
EF_DB_KEY_STORE=${EF_CONF_ROOT}/enginframe/certs/db.keystore
```

To migrate the credentials of your external database from the configuration file to the keystore:

1. Edit the `${EF_CONF_ROOT}/enginframe/server.conf` file as follows.
   a. Remove the ef.db.admin.name and ef.db.admin.password entries.

      Note
      Before deleting these entries, make a note of the value that they are set to. You need them for the following steps.

   b. Add or modify the `EF_DB_KEY_STORE_ENABLED` entry. Set it to **true**.
   c. Add an entry for

      `EF_DB_KEY_STORE=${EF_CONF_ROOT}/enginframe/certs/db.keystore`.
2. Run the following command.

   `${EF_TOP}/bin/ef-db-credentials.sh --action setDBAdmin --name` *name* `-- password` *password*

   Set *name* and *password* to the values that `ef.db.admin.name` and ef.db.admin.password were set to in the `${EF_CONF_ROOT}/enginframe/server.conf` file in Step 1.

3. Restart EF Portal. For more information, see Running NI SP EF Portal.

Configure EF Portal to work behind a network proxy

Learn how to provide the proxy information EF Portal needs to route network calls correctly. You'll use two java properties:

```
-Dhttp.proxyHost=PROXY HOST
-Dhttp.proxyPort=PROXY PORT
```

If the proxy is configured to work with HTTPS, use:

```
-Dhttps.proxyHost=PROXY HOST
-Dhttps.proxyPort=PROXY PORT
```

Set the properties in EF Portal

1. Get the java properties that EF Portal currently uses:
   a. Run `ps -aux` and locate the EF Portal process that ends in `org.apache.catalina.startup.Bootstrap start`.
   b. Copy the line representing the command that launched EF Portal and copy it to a local bash variable, such as OPTIONS:

```
$ OPTIONS="content of copied line"
OPTIONS_UPDATED=`echo "$OPTIONS" | tr ' ' '\n' | egrep -e '^-D' |
tr '\n' ' '`
```

The new variable OPTIONS_UPDATED has the content you need to use when you edit the EF Portal.conf file.

2. Edit the EF Portal.conf file by adding the variable name `SERVER_JAVA_OPTIONS` to the end of the file and saving it:

```
# other variables defined
...
SERVER_JAVA_OPTIONS="<values copied from the content of
OPTIONS_UPDATED> -
Dhttp.proxyHost=$PROXY_HOST -Dhttp.proxyPort=$PROXY_PORT"
```

`PROXY_HOST` and `PROXY_PORT` are placeholders for the proxy host and proxy port.

3. Restart the EF Portal process.

Running the NI SP EF Portal

This chapter describes how to start and stop EF Portal Portal and check its status. It also covers administration monitoring and self-check services.

Start or stop EF Portal and check its status

You can control EF Portal with the `$EF_TOP/bin/enginframe` command line script.

Run the following command to start EF Portal:

```
# $EF_TOP/bin/enginframe start
```

Run the following command to stop EF Portal:

```
# $EF_TOP/bin/enginframe stop
```

Note
If EF Portal Server and EF Portal Agent are on separate hosts, run the following command on the host that's running EF Portal Server:

```
# $EF_TOP/bin/enginframe <start|stop> server
```

Run the following command on the host that's running EF Portal Agent:

```
# $EF_TOP/bin/enginframe <start|stop> agent
```

Note
To start EF Portal Enterprise, run the EF Portal start (stop) command on each host of the EF Portal Enterprise infrastructure that's in your deployment. For more information, see Deployment.

```
# $EF_TOP/bin/enginframe <start|stop>
```

This control script also checks EF Portal's status:

```
# $EF_TOP/bin/enginframe status
```

The output of the previous command is as follows:

Start or stop EF Portal and check its status

```
# /opt/nisp/enginframe/bin/enginframe start

Reading EF Portal version from:
/opt/nisp/enginframe/current-version
Current version: 2025.2-r2056

EF Portal Control Script
Loading configuration from:
 - "/opt/nisp/enginframe/conf/efportal.conf"
Using EF Portal in "/opt/nisp/enginframe/2025.2-r2056"
Tomcat started.

[OK] EF Portal Server started

[OK] EF Portal Agent started
```

```
# /opt/nisp/enginframe/bin/enginframe status
Reading EF Portal version from:
/opt/nisp/enginframe/current-version
Current version: 2025.2-r2056

EF Portal Control Script
Loading configuration from:
 - "/opt/nisp/enginframe/conf/efportal.conf"
Using EF Portal in "/opt/nisp/enginframe/2025.2-r2056"

---- Server PID Information ----
USER PID PPID %CPU %MEM STIME TIME COMMAND
efnobody 1674 1 69.9 17.4 19:34 00:01:47 /usr/lib/jvm/jre/bin/java
-Xms1024m -Xmx1024m
-XX:HeapDumpPath=/[...]/dumps/server.pid1549.hprof
-Djava.protocol.handler.pkgs=com.efportal.common.utils.xml.handlers
-XX:ErrorFile=/[...]/dumps/server.hs_err_pid1549.log
-DjvmRoute=efserver1 -DEF_LICENSE_PATH=/opt/nisp/enginframe/license
-DDERBY_DATA=/opt/nisp/enginframe/data/derby
-DEF_ERRORS_DIR=/opt/nisp/ efportal/data/errors
-Def.repository.dir=/opt/nisp/enginframe/repository
-XX:+HeapDumpOnOutOfMemoryError
-DEF_ROOT=/opt/nisp/enginframe/2025.2-r2056/ efportal
-DEF_DYNAMIC_ROOT=/opt/nisp/enginframe/2025.2-r2056/efportal
-DEF_CONF_ROOT=/opt/nisp/enginframe/conf
-DEF_DATA_ROOT=/opt/nisp/enginframe/ data
-Def.tmp.dir=/opt/nisp/enginframe/tmp/efserver1
-DEF_SPOOLER_DIR=/opt/nisp/enginframe/spoolers
-DEF_SESSION_SPOOLER_DIR=/opt/nisp/enginframe/spoolers
-DEF_LOGDIR=/opt/nisp/enginframe/logs/efserver1
-Dfile.encoding=UTF -classpath
:/opt/nisp/enginframe/2025.2-r2056/tomcat/lib/
```

```
sdftree-handler.jar:/opt/nisp/enginframe/2025.2-r2056/tomcat/bin/
bootstrap.jar:/opt/nisp/enginframe/2025.2-r2056/tomcat/bin/tomcat-j
uli.jar   [...] org.apache.catalina.startup.Bootstrap start

---- Server Port Information ----
INFO: Starting ProtocolHandler ["http-bio-8443"]

---- Agent PID Information ----
root 1677 1 6.5 5.2 19:34 00:00:10 /usr/lib/jvm/jre/bin/java
-Xms512m -Xmx512m -XX:HeapDumpPath=/[...]/dumps/agent.pid1549.hprof
-XX:ErrorFile=/[...]/dumps/agent.hs_err_pid1549.log
-DEF_ROOT=/opt/nisp/enginframe/2025.2-r2056/efportal
-DEF_DYNAMIC_ROOT=/opt/nisp/enginframe/2025.2-r2056/efportal
-DEF_CONF_ROOT=/opt/nisp/efportal/conf
-DEF_DATA_ROOT=/opt/nisp/enginframe/data -DEF_SPOOLER_DIR=
/opt/nisp/enginframe/spoolers
-DEF_SESSION_SPOOLER_DIR=/opt/nisp/enginframe/spoolers
-DEF_LOGDIR=/opt/nisp/enginframe/logs/efserver1
-Dfile.encoding=UTF-8
-Djava.security.policy==/[...]/2025.2-r2056/efportal/conf/ef_java.p
olicy   [...] -jar
/opt/nisp/enginframe/2025.2-r2056/efportal/agent/agent.jar
```

```
# /opt/nisp/enginframe/bin/enginframe stop

Reading          EF          Portal          version          from:
/opt/nisp/enginframe/current-version
Current version: 2025.2-r2056
EF Portal Control Script
Loading configuration from:
 - "/opt/nisp/enginframe/conf/efportal.conf"
Using EF Portal in "/opt/nisp/enginframe/2025.2-r2056"
Tomcat stopped.

[OK] EF Portal Agent is down
```

Accessing the portal

After the EF Portal daemons are running, you can access EF Portal Portal in a browser window. To do this, in the browser's address bar, enter the host name of your EF Portal Server followed by a colon (:) and your EF Portal Server port number.

The EF Portal Server port number was set during installation and can be viewed with the following command.

```
# $EF_TOP/bin/enginframe status
```

For example, if the EF Portal Server host is named *myhost*, and EF Portal Server port number is 7070, type in your browser's address bar as follows:

```
http://myhost:7070
```

If your host name (in this example, myhost) isn't resolved by your DNS, you can specify the corresponding IP address:

`http://192.168.0.10:7070`

Note
If you encounter an issue related to the DNS name, domain resolution, or the IP address, contact your network administrator for help.

If EF Portal Server is installed successfully, the welcome page is displayed when you access the portal. If your browser reports errors such as Cannot find the requested page, Server not found, or Problem loading page, verify that EF Portal is installed correctly by checking its status as described in Start or stop EF Portal and check its status.

EF Portal roles

The roles assigned to the EF Portal host are determined during installation. They can be adjusted manually by setting the `EF_ROLES` parameter in the `$EF_TOP/conf/enginframe.conf` file.

```
# Comma separated list of roles enabled for the EF Portal.
# The roles assigned to the EF Portal host are determined during
installation.
# Each role corresponds to a component that can be activated.
# - server : can be removed only if the host runs only the EF Portal
Agent and not the EF Portal Server
# - agent  : must be set when the EF Portal Agent runs on the same
host as the server (common case)
# - derby  : included and required only if the Derby database was
selected during installation
# Default: server,agent,derby
#EF_ROLES=server,agent,derby
```

Additional roles can also be enabled:

```
# Comma separated list of additional roles for EF Portal.
# Each role corresponds to a feature or component that can be
activated.
# - shell  : required to enable the Web Terminal feature
# - proxy  : required to enable the Proxy feature
# Example: shell,proxy
# Default: shell
#EF_ADDITIONAL_ROLES=shell,proxy
```

The `enginframe` command manages the startup and shutdown of the enabled components. You can start or stop individual features by specifying them explicitly:

```
# $EF_TOP/bin/enginframe <start|stop> shell
```

```
# $EF_TOP/bin/enginframe <start|stop> proxy
```

For details on the Web Terminal (shell) and Web Proxy (proxy), refer to their respective sections.

Demo sites

If you installed the EF Portal *Developer's Documentation* when you installed EF Portal, the

welcome page, together with the production portal Applications, Views, and Operational Dashboard, displays a link to the Technology Showcase, which points to a set of demo services. These demo services provide an illustration of all of EF Portal's service capabilities.

Important
By default, only the `EF_ADMIN` user can access administration and tutorial demo sites.

Operational Dashboard

In the EF Portal *Operational Dashboard* view, administrators can monitor and manage operations directly from a web browser.

The Operational Dashboard is linked from the EF Portal welcome page. Otherwise, you can reach it directly at:

`http://host:port/context/admin`

The Operational Dashboard offers a set of services that are divided into the following categories:

Topics

- Monitor
- Troubleshooting
- EF Portal statistics

These services are listed in the navigation pane of the dashboard.

Monitor

The **Monitor** section provides tools for observing and managing the operational state of EF Portal.

- **Dashboard** displays a real-time overview of EF server load, Java™ Virtual Machine usage, license status, storage usage, active sessions, and running jobs.
- **Server Load** shows detailed metrics including CPU usage, JVM memory usage, repository and spooler file system size, and i-node usage.
- **Usage Statistics** presents current and historical data on logged-in users, jobs with their statuses, and interactive sessions and spoolers.
- **License Status** shows how your licenses are being utilized, including historical usage charts.
- **Installed Components** lists all installed EF Portal plugins along with their versions.
- **Release Notes** documents all changes made to EF Portal across versions. You can browse entries by category (New Features, Enhancements, Fixes, and Changes), and filter by EF Portal version or category to quickly find relevant updates.
- **Logged Users** lists all users currently logged into the portal, with the option to force a logout for individual users.
- **Triggers l**ets you view and manage scheduled triggers in EF Portal.
- **ACL Actors** lists the EF Portal ACL actors defined in the `authorization.xconf` files loaded by the system.

Troubleshooting

The **Troubleshooting** section provides tools for diagnosing and resolving issues with EF Portal.

- **Run Self Checks** performs a series of automated tests covering different functions and aspects of EF Portal. Each test returns a result and, in most cases, a suggested fix for any detected problem. The service can also be used to verify changes to system files before

performing an EF Portal version upgrade.

- **View Log Files** lets administrators inspect EF Portal log files directly in the browser. It includes a themed viewer, in-file search with navigation, tail mode with auto-refresh, log level color coding, multi-host support, and cross-file keyword search with CSV export.
- **Collect Support Info** gathers comprehensive information about EF Portal and its configuration, and packages it as a compressed archive. Attach this archive when submitting a request to EF Portal support.
- **NI SP Support** links to the NI SP support email and website, and provides access to the NI SP Support Knowledge Base AI Q&A for instant answers to common questions.

Advanced Administration

The **Advanced Administration** section provides tools for low-level maintenance and diagnostics of EF Portal internals.

- **View Job Cache Errors** displays errors that occurred during interactions between the Job Cache and the scheduler. Use this when diagnosing and resolving Job Cache-related issues.
- **Show Job Cache Content** provides a complete view of the Job Cache database contents. Use this when diagnosing and resolving Job Cache-related issues.
- **Clean Job Cache Errors** cleans the job cache errors database.
- **Delete Job Cache Content** permanently deletes all content from the Job Cache database. Any historical data for jobs no longer present in the scheduler will be irreversibly lost.
- **Cleanup Interactive Sessions** removes VDI sessions that are in an inconsistent state and cannot be automatically cleaned up by the portal.

EF Portal statistics

To collect information about license usage, jobs usage, and other useful statistics, EF Portal uses RRD4J as a round-robin database.

RRD4J is a high-performance data logging and graphing system for time series data, implementing RRDTool's functionality in Java™. It follows much of the same logic and uses the same data sources, archive types, and definitions that RRDTool uses.

EF Portal creates a database for general usage information that's named efstatistics.rrd and a database for each license file that's named `license_<component>_<expiration>_<maxToken>.rrd`. A new database will be created when a license file changes.

In the `admin.statistics.efconf` configuration file, you can configure some RRD4J specific parameters to change archive time intervals or to configure historical charts. For more information, see the RRD4J website.
When you run EF Portal for the first time, database files are created and loaded with data, and then they're updated every 60 seconds with new data.

HPC Workspace

In the EF Portal *HPC Workspace* view, users can create, manage, and submit both batch and interactive services.

The HPC Workspace is linked from the EF Portal welcome page or can be reached directly using the following URL.

`http://<host>:<port>/<context>/applications`

The HPC Workspace offers two interfaces for two different user roles: Admin View and User View.

Admin View

You can use the *Admin View* interface to monitor interactive sessions, jobs, hosts, and to manage services, HPC Workspace users and portal appearance.

- *Monitor » All Sessions* can be used by the administrator to manage interactive sessions for all Workspace users.
- *Monitor » All Jobs* can be used by the administrator to monitor and manage DRM jobs for all Workspace users. To control the jobs of other users' HPC Workspace administrators must have the proper rights in the underlying DRM.
- *Monitor » Hosts* can be used by the administrator to monitor the status of the hosts of the configured DRMs.
- *Manage » Services* can be used by the administrator to create, delete, edit, or publish batch and interactive services.
- *Manage » Users* service allows the administrator to register, import, or manage HPC Workspace users.
- *Manage » Notification* service allows the send custom notifications to all users in a role or to specific users.
- *Manage » Appearance* service enables the administrator to change the company logo and the Portal color theme in addition to custom CSS settings to style EF Portal according to the needs of e.g. the company with styling of colors, fonts, padding, etc. Starting from EF Portal 2025.2 it is possible to configure the Message of the Day (MOTD) to display to the users at login time.
- *Manage » Configuration* service allows administrators to manage EF Portal configuration, review the current settings, validate new changes, and apply modifications.
- *Manage » Operational Dashboard* opens the Operational Dashboard, where users can view system status, usage statistics, log files, and access tools to troubleshoot issues.

User View

You can use *User View* to monitor user data, sessions, jobs and hosts, together with Batch and Interactive services that are published by the HPC Workspace administrators.

- *Data » Spoolers* shows the user's EF Portal Spoolers and provides rename and delete operations.
- *Data » Files* allows the user to browse and manage files in their home directory.
- *Monitor » Sessions* can be used by the user to monitor and manage their interactive sessions.
- *Monitor » Jobs* can be used by the user to monitor and manage their jobs.
- *Monitor » Hosts* can be used by the user to monitor the status of the hosts of the configured DRMs.

HPC Workspace administrators can create and publish new services through the Admin View. When they publish a new service, administrators can make it available to all users or only to specific groups of users.

Note
The HPC Workspace requires a specific license. Contact <support@ni-sp-software.com> or your EF Portal reseller to purchase a license, perform a license change or obtain a demo license. EF Portal doesn't require a license on an EC2 instance. For more information about licensing, see Obtaining NI SP EF Portal.

Virtual Desktop

EF Portal includes the Virtual Desktop view, to create, manage, and submit Interactive services. The Virtual Desktop is linked from the EF Portal welcome page or can be reached directly at:
`http://<host>:<port>/<context>/vdi`

The Virtual Desktop offers two interfaces for two different users' roles: Admin's Portal and User View.

Admin View

You can use the *Admin View* to monitor and manage interactive sessions, jobs, hosts, services, users and portal appearance.

- *Monitor » All Sessions* can be used by the administrator to manage interactive sessions for all Virtual Desktop users.
- *Monitor » Hosts* can be used by the administrator to monitor the status of the hosts of the configured DRMs.
- *Manage » Interactive Services* can be used by the administrator to create, delete, edit, or publish Interactive services.
- *Manage » Users* can be used by the administrator to register, import, and manage Virtual Desktop users.
- *Manage » Notification* service allows the send custom notifications to all users in a role or to specific users.
- *Manage » Appearance* can be used by the administrator to change the company logo and the Portal color theme in addition to custom CSS settings to style EF Portal according to the needs of e.g. the company with styling of colors, fonts, padding, etc. Starting from EF Portal 2025.2 it is possible to configure the Message of the Day (MOTD) to display to the users at login time.
- *Manage » Configuration* service allows administrators to manage EF Portal configuration, review the current settings, validate new changes, and apply modifications.
- *Manage » Operational Dashboard* opens the Operational Dashboard, where users can view system status, usage statistics, log files, and access tools to troubleshoot issues.

User View

You can use the *User View* to monitor user's sessions and cluster hosts, together with the Interactive services published by Virtual Desktop administrators.

- *Monitor » Sessions* can be used by the user to monitor and manage their interactive sessions.
- *Monitor » Hosts* can be used by the user to monitor the status of the hosts of the configured DRMs.

Virtual Desktop administrators can create and publish new services through the Admin View. They can publish these services to specific users or groups of users.

EF Portal REST API

EF Portal 2025.0 introduced a cutting-edge, feature-rich REST API designed to streamline and enhance your integration experience. This modern API replaces the legacy Web Services approach, which was previously supported by a Java-based `efclient`, with newer, more efficient technologies that align with industry standards.

Key Features of the EF Portal REST API:

- **JSON Output for Easy Consumption:** Simplified data exchange for seamless integration into scripts, command-line tools, and third-party applications.
- **Programmatic Job Submission:** Automate job submissions directly from scripts or the command line.
- **Enhanced Automation and Batch Operations:** Effortlessly manage repetitive tasks with robust automation capabilities.
- **Comprehensive Job Management:** Query job information, control job states (Cancel, Suspend, Resume), and monitor progress in real-time.
- **Cluster Node Insights:** Gather detailed information about cluster nodes for improved resource management.
- **File Management Capabilities:**
    - Upload files directly to EF Portal from scripts or the command line.
    - Monitor and download output files from HPC jobs efficiently.
- **User Monitoring:** Monitor logged-in users for better oversight and security and license usage analysis.

Why Use EF Portal REST API?

The EF Portal REST API is built on OpenAPI/Swagger standards, offering unparalleled ease of integration into workflows. It supports:

- Cross-scheduler and cross-platform compatibility for diverse environments.
- Real-time monitoring of jobs and resources.
- Seamless interaction with third-party tools based on industry-standard protocols.

With the REST API you can extend automation a step further and push the boundaries towards an end-to-end HPC automated workflow: from job submit to monitoring, to downloading results, to post-processing.

Developer-Friendly Features

With support for popular programming languages like Python, PHP, Node.js, Go, Rust, and more, developers can quickly generate EF Portal REST clients tailored to their needs. For simpler use cases, direct cURL access is also available.

EF Portal 2025.0 empowers users with simplified integration processes, enhanced automation capabilities, and real-time monitoring - all designed to optimize your HPC workflows and improve operational efficiency.

The new EF Portal client - **efpclient** - is implemented in Python and offers easy consumption of the REST API via an auto-completing client supporting all REST API features.

Below is an overview of EF Portal REST API endpoints and features.

API Structure

The EF Portal API is organized into three distinct categories for easy navigation and use:

1. HPC Schedulers: Manage hosts, jobs and queues.
2. Monitor: Access user information and manage licenses (admin-only operations).
3. System: Work with services, spoolers, and file operations.

Key Features

Grid Operations

- Host and queue management
    - List hosts
    - Get details of a host
    - Get details of a cluster in a scheduler
- Job control
    - List all jobs
    - List jobs belonging to a user
    - List jobs running on a specific host
    - Get details of a job
    - Delete a job
    - Resume a job
    - Suspend a job
    - Find a job related to a spooler
- VDI Management
    - Get Session Information
    - Connect
    - Show all Sessions
    - Session Information
    - Close Session

System Operations

- Create an REST API Access Token
- Service management
    - Submit a Service including file upload
    - List options of a service
- Spooler control and file management
    - List all Spoolers
    - Remove a Spooler
    - List files in a Spooler
    - Upload file into a spooler

Monitor Operations (*Admin Only*)

- User session management (List users logged in, logout a user)
- License control and usage monitoring

The REST API Documentation including the OpenAPI/Swagger `openapi.json` definition can be found at docs/rest/ in the EF Portal installation, e.g. https://efp:8443/enginframe/docs/rest/.

EF Portal Client - efpclient

Here you can find the documentation of the new EFPClient implemented in Python in PyPi and then download and install the efpclient: https://pypi.org/project/efpclient. The efpclient is also available as a downloadable package.

Here is an overview of typical EF Portal Client use cases:

- Create a New Authentication Token
- Get all Jobs on the default HPC Scheduler or VDI Session Manager
- Get Info for Job with id 142
- List all Services available for the user in the HPC Applications Portal
- Describe a Service available in the HPC Applications Portal
- Submit the job.submission Service Using local file `$HOME/file.log` and Compression Level 4
- List Files in Spooler after a Service Submission
- Download a File from the Spooler
- Upload a File into the Spooler
- List Spoolers

The EFPClient supports the same configuration file like the efclient (EnginFrame WS Client) in the past for backward compatibility and a new configuration file: `$HOME/.efpclient.conf`.

Here we show the commands of a typical use case in the shell or a script:

```
# Install the EFP Client
> python3 -m pip install efpclient
# pip install efpclient
# Start the client command
> efpclient
Commands:
  admin     Administration related commands group.
  clusters  Clusters related commands group.
  hosts     Hosts related commands group.
  jobs      Jobs related commands group.
  queues    Queues related commands group.
  services  Services related commands group.
  spoolers  Spoolers related commands group.
  token     Authentication token commands group.

# Get the REST API token
> efpclient token create --token-only
[https://demo.ni-sp.com:8448/enginframe]  Enter  the  username  to
authenticate: username
```

```
[https://demo.ni-sp.com:8448/enginframe] Enter the password to
authenticate:
20d00e78fd87cc6179e16e7b02fe1d1427ce340d
# Or in one line with the credentials in the config file only readable
for the user:
# export EFP_CLIENT_TOKEN=$(efpclient token create --token-only)

> efpclient jobs all
[
    {
        "manager": "slurm",
        "id": "85",
        "name": "Job_Compress_Image_078.png",
        "owner": "efadmin",
        "queue": "test",
        "total_cpu_usage": "0:01",
        "memory_usage": "0",
        "swap_usage": "0",
        "execution_hosts": "demo",
        "submission_time": "2025-04-01T11:13:53",
        "execution_time": "2025-04-01T11:13:53",
                                                "execution_directory":
"/opt/nisp/enginframe/spoolers/efadmin/tmp3093980369702684817.ef",
        "nice": "0",
        "reasons": [
            {
                "value": "None"
            }
        ],
        "status": {
            "ef": "Done",
            "grid": "COMPLETED",
            "value": "DONE"
        }
    }]

> efpclient hosts
Commands:
  all   List all available hosts.
  info  Get host info for a given host name.
  jobs  List jobs by host name, same as 'jobs host' command.

# Enable autocompletion
eval "$(_EFPCLIENT_COMPLETE=bash_source efpclient)"
efpclient hosts <tab>
all   info  jobs   # output from the auto completion

# Configure the HPC Applications SDF in $HOME/.efpclient.conf:
#                                 sdf                                  =
https://demo.ni-sp.com/enginframe/applications/applications.xml
```

```
# List the services available
> efpclient services list
[
    {
        "id": "batch_builtin_jupyter_notebook.published",
        "name": "Jupyter Notebook",
                                                                "uri":
"//applications/batch_builtin_jupyter_notebook.published"
    },
    {
        "id": "batch_builtin_sample_compress_job.published",
        "name": "Sample Compress Job",
                                                                "uri":
"//applications/batch_builtin_sample_compress_job.published"
    }
]
> efpclient services describe -s
//applications/batch_builtin_sample_compress_job.published
{
    "id": "batch_builtin_sample_compress_job.published",
                                                                "uri":
"//applications/batch_builtin_sample_compress_job.published",
    "name": "Sample Compress Job",
    "options": [
        {
            "id": "file",
            "label": "File to compress:",
            "type": "sfu",
            "option": "--opt-file FILENAME",
             "help": "File to compress:. Specify a single local file to
upload."
        },
        {
            "id": "level",
            "label": "Compression level:",
            "type": "list",
            "value": "9",
            "choices": [
                {
                    "value": "9",
                    "label": "maximum"
                },
                {
                    "value": "4",
                    "label": "medium"
                },
                {
                    "value": "1",
                    "label": "minimum"
```

```
                }
            ],
            "option": "--opt-level VALUE_1|VALUE_2|...",
              "help": "Compression level:. Default value is: 9. Valid
values to choose from are: 9, 4, 1"
        },
        {
            "id": "cluster",
            "label": "Execution cluster:",
            "type": "list",
            "value": "cluster:slurm",
            "choices": [
                {
                    "value": "cluster:slurm",
                    "label": "cluster"
                }
            ],
            "option": "--opt-cluster VALUE_1|VALUE_2|...",
                     "help": "Execution cluster:. Default value is:
cluster:slurm. Valid values to choose from are: cluster:slurm"
        }
    ],
    "actions": [
        {
            "id": "submit",
            "result": "text/xml"
        }
    ]
}

# Submit the compress service with file results-112.out
> efpclient services submit -s
//applications/batch_builtin_sample_compress_job.published --opt-file
results-112.out --opt-level 4
{
                                                                "uri":
"spooler:///opt/nisp/enginframe/spoolers/efadmin/tmp955910390737051918
8.ef",
                                                              "output":
"/enginframe/rest/system/services?_uri=//com.enginframe.system/show.sp
ooler&_spooler=spooler%3A%2F%2F%2Fopt%2Fnisp%2Fenginframe%2Fspoolers%2
Fefadmin%2Ftmp9559103907370519188.ef"
}

# Use the Spooler URI to show the files in the Spooler
> efpclient spoolers files -u
spooler:///opt/nisp/enginframe/spoolers/efadmin/tmp9559103907370519188.e
f
[
    {
```

```
        "path": "/",
        "vroot": "8b45963b96801790f315dff51718ca9cb23db834",
        "name": "results-112.out.gz",
        "type": "file",
        "modified": "2025-04-08T11:31:00+00:00",
                                                                    "url":
"https://demo.ni-sp.com/enginframe/download?file=/8b45963b96801790f315
dff51718ca9cb23db834//results-112.out.gz&_spooler=spooler:///opt/nisp/
enginframe/spoolers/efadmin/tmp9559103907370519188.ef&_size=51&_plugin
=fm",
        "size": {
            "value": 51,
            "unit": "bytes"
        }
    },
    {
        "path": "/",
        "vroot": "8b45963b96801790f315dff51718ca9cb23db834",
        "name": "results-87.out",
        "type": "file",
        "modified": "2025-04-08T11:31:00+00:00",
                                                                    "url":
"https://demo.ni-sp.com/enginframe/download?file=/8b45963b96801790f315
dff51718ca9cb23db834//results-87.out&_spooler=spooler:///opt/nisp/engi
nframe/spoolers/efadmin/tmp9559103907370519188.ef&_size=0&_plugin=fm",
        "size": {
            "unit": "bytes"
        }
    }
]

# Download a file from the Spooler
> efpclient spoolers download -u
spooler:///opt/nisp/enginframe/spoolers/efadmin/tmp9559103907370519188.e
f -f results-112.out.gz
Content saved to results-112.out.gz
```


The EF Portal REST API offers a powerful and comprehensive control of EF Portal via own API clients or the Python-based efpclient installable from PyPi or as download.

API Authentication

The REST API uses the standard HTTP Bearer authentication scheme, where each request is authorized by presenting a time-bound access token in the `Authorization` header.

You can obtain a new token using either the *eftoken* plugin bundled in EF Portal or `efpclient`.

- Visit:
  `https://<host>:<port>/<context>/eftoken`
  to access the *eftoken* plugin and use the *Create Token* service to create a new token for

the logged in user.
Select the “Output the token only” checkbox to display only the token.

- Use `efpclient token create` to create a new token. For more information, see the `efpclient` documentation.

You can also call the *Create Token* service to create a new token programmatically. For example, from an EF Portal custom service you can use the following JavaScript code to create a new token for later use. Note the use of the `tokenOnly` parameter to return only the token value and avoid output parsing.

```
jQuery.ajax({
    type: 'POST',
    url: '/' + jQuery.enginframe.rootContext +
         '/eftoken/eftoken.xml?_uri=//eftoken/eftoken',
    data: {
        'EF_OUTPUT_MODE': 'rest',
        'tokenOnly': 'true'
    }
})
.done((response) => {
    // response contains only the token
    let token = response;
    ...
})
.error((error) => {
    // Create token failed
    ...
});
```

EFToken Configuration

The *eftoken* plugin generates time-bound access tokens for EF Portal REST API authentication. These tokens can be managed through the Web Configuration editor (EF Portal 2026.0) or by editing `$EF_CONF_ROOT/plugins/eftoken/properties.conf`.

EFTOKEN_TTL — Time validity duration for generated tokens. Default: 1h
Format: Combined duration string using d, h, m, s (days, hours, minutes, seconds)

Examples:

```
EFTOKEN_TTL=10d3h4m50s    # 10 days, 3 hours, 4 minutes, 50 seconds
EFTOKEN_TTL=30m           # 30 minutes
EFTOKEN_TTL=2h30m         # 2 hours, 30 minutes
```

EFTOKEN_AUTOMATIC_UPDATE — Controls automatic token renewal on login.

- true Token validity extends automatically on each successful login
- false Token expires based on `EFTOKEN_TTL`; login does not extend validity

Default: `false`

REST API Output JSON converted from XML

In addition to showing XML output from the REST API we automatically output JSON converted from XML in the REST API Output for easy consumption in other services. Here is an example of calling a service to get the connection file data which is by default XML:

```
curl --silent -k -X 'POST' \
  'https://efp.example.com:8443/enginframe/rest/system/services' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer 9a15aa62447f4e5dc8b23068a55c6f6ab9ec199e'\
  -H 'Content-Type: application/json' \
          -d     '{"sdf":     "/hydrogen/ui.hydrogen.xml",     "uri":
"//ui.hydrogen/interactive.connection.data",   "options":   [{   "name":
"sessionUri",                                                  "values":
["spooler:///opt/nisp/enginframe/spoolers/efadmin/tmp10801419285751316
539.session.ef"]}]}' \
| jq -r .outputJson | jq
```

Example output of the JSON output of the connection file data service:

```
{
  "connection": {
    "param": [
      {
        "name": "connectFileUri",
        "content": "//com.enginframe.interactive/generate.dcv2file"
      },
      {
        "name": "connectFileExt",
        "content": ".dcv"
      },
      {
        "name": "uriScheme",
        "content": "https"
      }, ……
```

Administration

Topics

Common administration tasks

Most of the tasks an EF Portal administrator has to perform involve editing configuration files. This chapter provides an overview of the main EF Portal configuration files and then focuses on some common administration tasks, explaining in detail how to accomplish them.

This chapter describes the following tasks:

Topics

Other important administration tasks regarding EF Portal Portal's specific sub-components (like spooler management, logging, etc.) are described in the next chapters.

Note
Starting March 31, 2022, NI SP EF Portal doesn't support VNC®, HP® RGS, VirtualGL, and DCV 2016 and previous versions. Integrations still might be working but are not supported.

Web-based Configuration Editor

Most EF Portal administration tasks involve changing configuration settings, typically by editing text files and managing overrides.

Starting from version 2026.0, EF Portal supports a web-based configuration editor that lets administrators review and change supported configuration parameters through the UI.

The configuration editor is available in both HPC Workspace (Applications) and Virtual Desktop's (Views) administration portals.

When to use it

Use the configuration editor when you need to change a supported parameter and want a guided workflow (searchable parameters, validation checks, grouped layout) instead of manually editing files.

The Configuration Editor also allows you to validate scheduler settings and verify the format of individual parameters.

Use manual file editing for advanced customization, or when you need to modify files that are not managed by the configuration editor.

What gets saved and where

When saving from the editor, EF Portal writes the corresponding overrides in the standard configuration override location (`$EF_TOP/conf` folder) so changes are preserved across updates.

In addition, EF Portal writes an audit entry for each save operation so you can later review what changed and eventually restore previous values. Audit files are available in the `${EF_DATA_ROOT}/plugins/config-manager/save-history` folder.

Restart-required changes

Some parameter changes require a server restart; the editor will warn you when your saved changes include restart-required parameters and will indicate that they may not be effective until EF Portal is restarted.
After a restart, the editor will automatically clear the "pending restart" status when it detects that the server has been restarted since the save.

Main configuration files

In this section the main EF Portal configuration files are described. Further details can be found throughout this guide.

Starting from EF Portal 2015, configuration files are isolated from the rest of EF Portal installation in `$EF_TOP/conf`. Configuration files in `$EF_TOP/conf` are preserved during updates.

EF Portal still uses internal configuration files located under `$EF_TOP/<VERSION>` and organized according to the pre-EnginFrame 2015 directory-tree layout (i.e. `$EF_TOP/<VERSION>/enginframe/conf` and `$EF_TOP/<VERSION>/enginframe/plugins/<plug-in>/conf`, ...).

Some of these files define default values which can be overridden using files with the same name under the `$EF_TOP/conf` tree. Note that any modifications to files under `$EF_TOP/<VERSION>` are discouraged and the files under this directory are subject to change in the next EF Portal versions without notice.

`enginframe.conf`

It is located in the `$EF_TOP/conf` directory.

As already seen in Fine-tuning your installation, this file configures the JDK running EF Portal Server and EF Portal Agent and the execution options passed to the JVM. It also configures other execution environment parameters like locale or user running Tomcat® (referred to as EF_NOBODY).

`server.conf`

This is the server's main configuration file.

It is located in the `$EF_TOP/<VERSION>/enginframe/conf` directory. Its contents are merged with `$EF_TOP/conf/enginframe/server.conf` if present. In case the same property is defined in both files the latter wins.

It also contains some parameters used by the *local agent* when executing services on EF Portal Server's host on `EF_NOBODY`'s behalf.

`agent.conf`

This is the agent's main configuration file.

It is located in the `$EF_TOP/<VERSION>/enginframe/conf` directory. Its contents are merged with `$EF_TOP/conf/enginframe/agent.conf` if present. In case the same property is defined in both files the latter wins.

`mime-types.xml`

It associates content types to files downloaded through the portal without requiring any change to the JDK settings.

It is located in the `$EF_TOP/<VERSION>/enginframe/conf` directory. Its contents are merged, extended or overridden, with `$EF_TOP/conf/enginframe/mime-types.xml` if present. In the case the same MIME type is defined in both files the latter overrides the mapping.

For more information, see Managing internet media types.

`log.server.xconf`

It configures EF Portal Server's logging.

It is located in the `$EF_TOP/<VERSION>/enginframe/conf` directory. Overridden by `$EF_TOP/ conf/enginframe/log.server.xconf` if present.

`log.agent.xconf`

It configures EF Portal Agent's logging.

It is located in the `$EF_TOP/<VERSION>/enginframe/conf` directory. Overridden by `$EF_TOP/ conf/enginframe/log.agent.xconf` if present.

`authorization.xconf`

It's a configuration file for the EF Portal authorization system. It defines users' groups and access control lists (ACLs).

Refer to Configuring authorization.

Deploying a new plugin

Two types of plugins exist from a deployment point-of-view: the official ones distributed by NI SP and the custom ones produced in-house or distributed by third parties.

Important
Plug-ins designed for pre-2015 EnginFrame versions cannot be installed on newer EF Portal. Please contact NI SP and check if updated plug-ins are available.

Official NI SP plugins

NI SP's official plugins are distributed with an installer that sets up the plugin and deploys it inside EF Portal.

If NI SP *jobhist* plugin (shipped separately) were to be installed, it would be done by executing:

```
# java -jar jobhist-X.Y.Z.jar
```

The installer asks EF Portal's root directory, plugin specific configuration options, and then installs the code.

If EF Portal Server and EF Portal Agent are installed on two different hosts, unless otherwise specified in plugin documentation, the plugin has to be installed on both hosts.

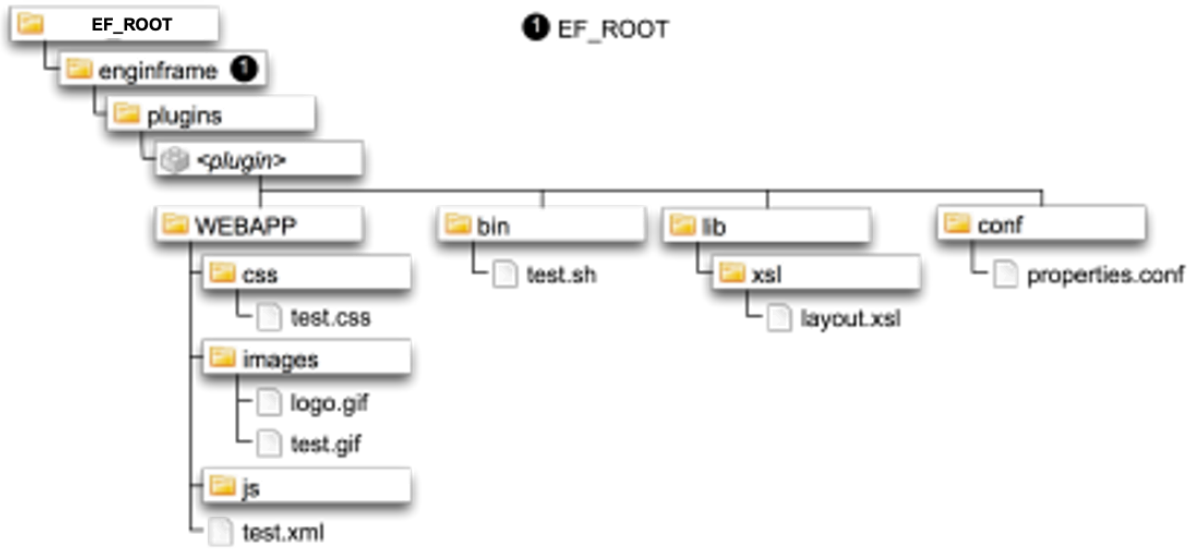
Unless stated otherwise in plugin documentation, once installed, the plugin is immediately available through EF Portal Portal without requiring a restart.

Custom plugins

If you have a custom plugin or are deploying a third party plugin that is not distributed with an installer, a manual deployment is necessary. All EF Portal plugins *must be* placed inside `$EF_TOP/<VERSION>/plugins` and follow the internal structure described below.

A plugin directory structure example is:

EF Portal Plugin Structure



Some plugins may need additional setup operations to work properly. Read and follow the instructions distributed with plugin documentation.

Changing Java™ version

In some cases, such as an important security fix shipped by a Java™ vendor, changing the Java™ Platform running EF Portal Portal is necessary.

The Java™ Platform running EF Portal Server and EF Portal Agent is defined using the `JAVA_HOME` parameter inside `$EF_TOP/conf/enginframe.conf`. Changing this value is the only step necessary to use a different Java™ version to run these components.

Using Java™ installed in the directory `/opt/java` is done as follows:

```
JAVA_HOME="/opt/java"
```

EF Portal Server and EF Portal Agent restart is necessary to make changes effective. Refer to

Java™ platform for further information on supported Java™ versions.

Changing the default agent

A single EF Portal Server can connect to many EF Portal Agents to execute services.

The remote *default agent* is specified during installation. When both server and agent are installed on the same host, this agent automatically becomes the default one. When the server is installed alone, the installer asks the default agent hostname and TCP port.

The default agent is defined in server.conf with two properties:

- `EF_AGENT_HOST`

  Specifies default Agent hostname or IP address. The default value is localhost.

- `EF_AGENT_PORT`

  Specifies default TCP port where Agent listens for incoming connections (i.e. parameter ef.agent.port in Agent configuration, see Configuring agent ports for more details). The default value is 9999.

It is located in the `$EF_TOP/<VERSION>/enginframe/conf` directory. Its contents are merged with `$EF_TOP/conf/enginframe/server.conf` if present. In case the same property is defined in both files the latter wins.

Service Definition Files use both properties.

So if an EF Portal Server wants to set its default agent as agent1 listening on port 7777 the parameters mentioned above become:

```
EF_AGENT_HOST=agent1
EF_AGENT_PORT=7777
```

`EF_AGENT_HOST` can also be set to an IP address, e.g. 192.168.1.16.

Changes to these parameters do not require EF Portal Server's restart.

Important
You *cannot* remove `EF_AGENT_HOST` and `EF_AGENT_PORT` from `server.conf`. If those properties are empty/missing, EF Portal uses the default values.

Managing internet media types

When a file is downloaded from EF Portal Portal the browser tries to open it with the application configured to manage its media type content. For example, if an image is downloaded the browser displays it inline, while if a Word document is downloaded, it launches Office.

EF Portal suggests to browsers the best method to handle files by sending the *Internet media type* in the download response.

An Internet media type, originally called *MIME type*, is a two-part identifier for file formats on the Internet. The identifiers were originally defined in RFC 2046 for use in e-mails sent through SMTP, but their use has expanded to other protocols such as HTTP.

It is very useful to link uncommon file extensions to specific MIME types, so browsers can handle them correctly. Not all browsers use MIME type information in the same way. Older *Microsoft® Internet Explorer®* versions rely more on content type detection provided by *Windows® operating system* than on MIME type specified by the server's response message, especially for some of the most common file name extensions.

EF Portal uses a built-in list of MIME types provided by the JDK. This list is defined in `JAVA_HOME/jre/lib/content-types.properties`.

EF Portal overrides and extends this list of MIME types with `$EF_TOP/<VERSION>/enginframe/conf/mime-types.xml` in the static installation directory and with the optional `$EF_TOP/conf/efportal/mime-types.xml` in the EF Portal custom configuration directory tree. These files, owned by EF Portal administrator, are used across the whole system unless more specific settings are found.

Each plugin has the possibility to extend and overrides the EF Portal MIME types settings, by defining its own static `$EF_TOP/<VERSION>/enginframe/plugins/`*`plugin_name`*`/conf/mime-types.xml` and the associated customizable version `$EF_TOP/conf/plugins/`*`plugin_name`*`/mime-types.xml`
in the EF Portal configuration directory tree. MIME types defined in the plugin mime-types.xml files are used when downloading files from spoolers generated by services defined in the *plugin_name* plugin.

When EF Portal receives a download request from the browser, it tries to associate a MIME type to the file. It first looks in the plugin specific mime-types.xml file, then in EF Portal mime-types.xml. In the case it cannot associate a MIME type from its own configuration files, it uses the default

definitions specified in JDK's content-types.properties.

More in details, when looking for a MIME type of a file EF Portal checks resources in the following order:

1. Custom plug-in MIME types,
   `$EF_TOP/conf/plugins/plugin_name/mime-types.xml`
2. Static plug-in MIME types,
   `$EF_TOP/<VERSION>/enginframe/plugins/plugin_name/conf/mime-types.xml`
3. Custom EF Portal system-wide MIME types,
   `$EF_TOP/conf/plugins/plugin_name/mime-types.xml`
4. Static EF Portal system-wide MIME types,
   `$EF_TOP/<VERSION>/enginframe/conf/mime-types.xml`
5. Default JDK MIME types, `JAVA_HOME/jre/lib/content-types.properties`

If this chain of lookup for a specific MIME type fails, the default MIME type, if defined, is returned, otherwise an empty MIME type is sent back in the HTTP response by EF Portal.

EF Portal Server dynamically reloads the mime-types.xml files when they are modified, so no restart is necessary.

EF Portal ships this `$EF_TOP/<VERSION>/enginframe/conf/mime-types.xml`:

```
<?xml version="1.0"?>
<ef:mime-types xmlns:ef="http://www.enginframe.com/2000/EnginFrame">

  <ef:default type="text/plain" />

  <ef:mime-type>
    <ef:type type="text/plain"/>
    <ef:desc desc="ASCII Text File"/>
    <ef:extensions>
      <ef:extension ext=".log"/>
    </ef:extensions>
    <ef:match-list>
      <ef:match expr="[A-Z0-9]*.ef" />
      <ef:match expr="README" casesensitive="false" />
    </ef:match-list>
  </ef:mime-type>
  <ef:mime-type>
    <ef:type type="application/x-javascript"/>
    <ef:desc desc="JavaScript File"/>
    <ef:match-list>
      <ef:match expr="^.*[Jj][Ss]" />
    </ef:match-list>
  </ef:mime-type>

  <ef:mime-type>
    <ef:type type="application/json"/>
    <ef:desc desc="JSON File"/>
    <ef:match-list>
      <ef:match expr="^.*\.[Jj][Ss][Oo][Nn]" />
    </ef:match-list>
  </ef:mime-type>

  <ef:mime-type>
```

```
    <ef:type type="text/css"/>
    <ef:desc desc="CSS File"/>
    <ef:match-list>
      <ef:match expr="^.*[Cc][Ss][Ss]" />
    </ef:match-list>
  </ef:mime-type>

  <ef:mime-type>
    <ef:type type="image/vnd.microsoft.icon"/>
    <ef:desc desc="ICO File"/>
    <ef:match-list>
      <ef:match expr="^.*[Ii][Cc][Oo]" />
    </ef:match-list>
  </ef:mime-type>
</ef:mime-types>
```

The `<ef:extensions>` section contains exact matches. Thus, in this example, EF Portal associates text/plain MIME type to any file ending with extension .log or .patch.

The `<ef:match-list>` section contains regular expressions for matching a filename. Thus, in this example, EF Portal associates text/plain MIME type to all files whose name contains only alphanumeric characters and whose extension is .ef and to all files named README.

Since the casesensitive attribute is set to false in the first `<ef:match>` tag, a matching is performed that is not case sensitive. This means, for example, that files named license.ef or LICENSE.EF are matched. If casesensitive attribute is not explicitly set to false, a case sensitive match is performed. So files named readme or ReadMe are not matched by the regular expression defined in the second `<ef:match>` tag.

`<ef:default>` is a child of `<ef:mime-types>` tag. It specifies a default MIME type for those cases where a MIME type cannot be guessed. The syntax is: `<ef:default type="`*`expected_mime_type`*`" forward-guess="`*`[true|false]`*`" />`

Attribute forward-guess set to true allows to interrupt MIME type lookup at the current mime types.xml file without considering upstream MIME type settings. Its default value is false.

The default value can be overridden following the same lookup order for mime-types.xml files as reported in the list above.

Important
There are two settings concerning security and MIME types in the server.conf configuration file that are important to be described here: `ef.download.mimetype.mapping.text` and `ef.download.mimetype.mapping.octetstream`.

`ef.download.mimetype.mapping.text`: a comma separated list of MIME types that, for security reasons, are mapped to text/plain in the HTTP response to clients when downloading a file. This further MIME type mapping prevents browsers from interpreting and rendering the downloaded files protecting against malicious code that could be executed on the client browser (cross-site scripting attack).

`ef.download.mimetype.mapping.octetstream`: a comma separated list of MIME types that, for security reasons, are mapped to application/octet-stream in the HTTP response to clients when downloading a file. This further MIME type mapping prevents browsers from taking any action on the downloaded files protecting against malicious code that could be executed on the client host.

Define custom extensions for text and image mime types

To configure the text and image preview in the file manager you can define custom extensions for text and image mime types.

Using the conf file `EF_ROOT/plugins/fm/conf/fm.conf` you can specify a custom colon separated list of extensions for both `text/*` and `image/*` mime types to let the File Manager handle them appropriately - e.g. enabling preview for image files or preview/edit/stream for text files.

You can also apply the conf at global level (so for all EF Portal versions installed) using the file `EF_CONF_ROOT/plugins/fm/fm.conf` (it does not exist by default today, you can just copy it from the versioned one)

New parameters in fm.conf are:

`FM_CUSTOM_EXTENSIONS_TEXT`

`FM_CUSTOM_EXTENSIONS_IMAGE`

The parameters are documented in the `fm.conf` file.

Customizing error page layout

Whenever EF Portal encounters an error during service execution, it displays an error message on the browser using a well known layout. All errors that end up on browsers are displayed with the same look and feel.

Error page layout customization is achieved by changing `ef.error.layout` value inside server.conf. This value must be an absolute path to an XSL file containing customized style sheets. `$EF_TOP/ <VERSION>/enginframe/lib/xsl/com.enginframe.error.xsl` is the default value for `ef.error.layout`. This file can be used as a starting point to create customized templates.

`$EF_TOP/<VERSION>/enginframe/plugins/acme/lib/xsl/mycompany.error.xsl` contains the customized XSL templates, then `ef.error.layout` has to be set as follows: `ef.error.layout=${EF_ROOT}/plugins/acme/lib/xsl/mycompany.error.xsl`

Changes to ef.error.layout do not require a server restart.

Limiting service output

EF Portal's usual client is a web browser. Limiting the amount of data sent to browsers saves resources on client-side. When a service execution produces a big amount of XML/HTML, the browser could have trouble rendering the page.

To avoid overloading the clients (and server/agent that have to produce/process the data), the maximum amount of data that services can produce is definable using ef.output.limit. If the limit is exceeded, the service's output is truncated and an error message is displayed on the browser.

`ef.output.limit` is specified as the number of bytes and the default value is 300 MB.

Since services are usually executed by a *remote agent*, this property is set inside the agent's agent.conf.

The following example shows how to set this property to limit service's output to 50 MB (52428800 bytes) of data: `ef.output.limit=52428800`

However, since *local agents* can also execute services, `ef.output.limit` is also defined inside

server.conf.

EF Portal Agent and/or the EF Portal Server restart is not required when changing this property.

Note
The service execution is not influenced in any way by the specified limit. The service output is truncated on the agent before sending it back to the server.

Configuring agent ports

EF Portal Agent and EF Portal Server communicate using Java™ RMI over SSL protocol. Technically speaking, EF Portal Agent is an RMI server that exposes a remote object whose methods are invoked by EF Portal Server.

For this reason, EF Portal Agent needs to open two TCP ports on its host: one port is used by an *RMI Registry* while the other one is used by an RMI server. These ports are chosen during installation (by default they are respectively 9999 and 9998).

`$EF_TOP/conf/enginframe/agent.conf` contains the values specified during installation. Edit this file to change these values:

- `ef.agent.port`

  Specifies TCP port on which RMI Registry is listening.

  If this property is empty, the default port 9999 is used.

  The specified value must be a valid TCP port that is not used by other processes on the same host.

- `ef.agent.bind.port`

  Specifies TCP port on which RMI server is listening.

  If this property is empty or is 0, a random free port is chosen at EF Portal Agent startup.

  The specified value must be 0 or a valid TCP port that is not used by other processes on the same host. Furthermore, the specified value must be different from `ef.agent.port`.

For example, using port 7777 for RMI Registry and port 7778 for RMI server, the two parameters must be set in the following way:

```
ef.agent.port=7777
ef.agent.bind.port=7778
```

EF Portal Agent must be restarted to make changes effective.

Tip
If there is a firewall between EF Portal Server and EF Portal Agent then `ef.agent.bind.port` has to be set to a value different from zero. The firewall has to be configured to allow EF Portal Server to open TCP connections towards EF Portal Agent using the ports specified by `ef.agent.port` and `ef.agent.bind.port`.

If `ef.agent.port` is changed, then all `<ef:location>`'s port attributes have to change accordingly inside *Service Definition Files*.

Modify EF_AGENT_PORT inside `$EF_TOP/conf/enginframe/server.conf` if the default agent's ports changed. Refer to Changing the default agent for more details.

Customizing user switching

Unless EF Portal was installed by an unprivileged user, every time EF Portal Agent runs a service it *impersonates* the system user associated with the portal user requesting service execution. This ensures service execution is performed as a regular system user (root is not allowed to run services) and the files created/modified have proper ownerships and permissions.

EF Portal allows modifying how *user switching* is done to affect service execution environment and ultimately service execution itself.

EF Portal Agent user switching mechanism is based on **su** shipped with every Linux®.

`$EF_TOP/conf/enginframe/agent.conf` configures `ef.switch.user.params` parameter used to specify options passed to **su**. Multiple parameters must be separated by a space without using quotes or double quotes like in the following example: `ef.switch.user.params=-f -m`

An empty ef.switch.user.params means no options are passed to **su**:
`ef.switch.user.params=`

A missing `ef.switch.user.params` is automatically set to - which results in the user's profile being sourced when **su** is executed. By default, `ef.switch.user.params` property is not set.

Tip
If user profiles on EF Portal Agent's host are complicated and sourcing them affects service execution performance, it is suggested to set `ef.switch.user.params` to avoid sourcing them when **su** is executed. You can, for example, set `ef.switch.user.params` to the empty string.

EF Portal Agent does not have to be restarted when changing this parameter.

Customizing user session timeout

A session defines a user's working period within EF Portal Portal. A session starts at user login and ends either when user logs out or when EF Portal Server invalidates it.

Session timeout specifies the number of minutes a user can remain idle before the portal terminates the session automatically. If a user does not interact with EF Portal within the configured timeout, the session is automatically invalidated and the user has to authenticate again to access EF Portal Portal.

The default session timeout, defined for all users, is set to *30 minutes*.

Session timeout can be changed in `EF_ROOT/WEBAPP/WEB-INF/web.xml` by changing the session timeout value. This value is expressed in a whole number of minutes. If the timeout is *0* or less, the container ensures the default behaviour of sessions is never to time out.

Changing session timeout to two hours, can be achieved modifying session-timeout value in the following way:

```
<session-config><session-timeout>120</session-timeout></session config>
```

Changes to session timeout require EF Portal Server's restart.

Apache®-Tomcat® connection

There are many reasons to integrate Tomcat® with Apache®. And there are reasons why it should not be done too. Starting with the newer Tomcat (EF Portal ships version 9.0.115), performance

reasons are harder to justify. So here are the issues to discuss in integrating vs not:

- *Encryption* - The Apache HTTP Server module mod_ssl is an interface to the OpenSSL library, which provides Strong Encryption using the Secure Sockets Layer and Transport Layer Security protocols. Tomcat is able to provide a similar encryption using the JVM, which needs to be cross platform, so it is somehow less efficient than Apache. Moreover, Apache has a longer experience in this field.
- *Clustering* - By using Apache as a front end you can let Apache act as a front door to your content to multiple Tomcat instances. If one of your Tomcats fails, Apache ignores it and your Sysadmin can sleep through the night. This point could be ignored if you use a hardware load balancer and the clustering capabilities of EF Portal Enterprise Edition.
- *Clustering/Security* - You can also use Apache as a front door to different Tomcats for different URL namespaces (/app1/, /app2/, /app3/, or virtual hosts). The Tomcats can then be each in a protected area and from a security point of view, you only need to worry about the Apache server. Essentially, Apache becomes a smart proxy server.
- *Security* - This topic can sway one way or another. Java™ has the security manager while Apache has a larger mindshare and more tricks with respect to security. Details will not be given here, but let Google™ be your friend. Depending on your scenario, one might be better than the other. But also keep in mind, if you run Apache with Tomcat you have two systems to defend, not one.
- *Add-ons* - Adding on CGI, Perl, PHP is natural to Apache. It's slower for Tomcat. Apache also has hundreds of modules that can be plugged in at will. Tomcat can have this ability, but the code has not been written yet.
- *Decorators* - With Apache in front of Tomcat, you can perform any number of decorators that Tomcat does not support or does not have immediate code support. For example, mod_headers, mod_rewrite, and mod_alias could be written for Tomcat, but why reinvent the wheel when Apache has done it so well?
- *Speed* - Apache is faster at serving static content than Tomcat. But unless you have a high traffic site, this point is useless. But in some scenarios, Tomcat can be faster than Apache. So benchmark *your* site.
- *Socket handling/system stability* - Apache has better socket handling with respect to error conditions than Tomcat. The main reason is that Tomcat must perform all its socket handling via the JVM which needs to be cross platform. The problem is that socket optimization is a platform specific ordeal. Most of the time the Java™ code is fine, but when you are also bombarded with dropped connections, invalid packets, invalid requests from invalid IPs, Apache does a better job at dropping these error conditions than a JVM based program. (YMMV)

[Source: Tomcat Wiki].

There are at least two ways to configure an Apache Web Server as a frontend to Tomcat according to the protocol used:

- HTTP
- AJP [see Protocol Reference].

The connection between Apache and Tomcat using protocol AJP can follow two different strategies:

- Apache Module mod_proxy_ajp (Apache version 2.2 or higher)
- Tomcat Connector JK

Changing charts backend

Charts can be embedded dynamically in EF Portal web pages.

By default, the internal charts provider is used, but is possible to use any other service compatible with Google™ Chart API.

The chart backend can be changed in two ways:

- Globally for all charts that EF Portal produces.
- Locally for specific charts.

In the first case, edit `$EF_TOP/conf/enginframe/server.conf` specifying ef.charts.base.url:

```
ef.charts.base.url=http://chart.apis.google.com/chart
```

In the second case, set base attribute inside chart root tag:

```
<ch:chart ... base="http://chart.apis.google.com/chart" ... >
```

Interactive administration

Topics

Configuration files

Most of the time the values defined during the Interactive Plugin installation provide all the information necessary to have a working setup. However sometimes further configuration is needed to tailor the session broker to specific system and network conditions or to change the values defined during the installation.

All the Interactive Plugin configuration files are located in the conf subdirectory.

Note
All the parameters in the configuration files with extension different from `.efconf` comply with the following format from Bourne shell: `PARAMETER_NAME="parameter value"` In particular,

- There are *no spaces* before and after the = (equals).
- You can use shell variable references with the usual syntax $variable. Always enclose variable names with curly braces, for example: `${HOME}`.
- Bourne shell escaping and quoting syntax apply. Be sure to enclose values containing spaces within the most appropriate quotes.

Important
Configuration parameters are automatically loaded upon saving. No need to restart EF Portal or logout.

Topics

- url.mapping.conf
- xstartup files
- mime-types.xml

interactive.efconf

This file contains Interactive Plugin's main default configuration parameters, that can be usually overridden by each portal service.

Default Parameters

INTERACTIVE_DEFAULT_OS

- value: *required*
- default: *linux*

By default, interactive sessions will be launched on the operating system stated by `INTERACTIVE_DEFAULT_OS` parameter.

Available values:

- linux - schedule on Linux® operating systems
- windows - schedule on Windows® operating systems

This behavior can be overridden by each service itself by using `--os <system>` option of *interactive.submit*

Example: `INTERACTIVE_DEFAULT_OS=linux`

INTERACTIVE_DEFAULT_JOBMANAGER

- value: *optional*
- default: the configured grid scheduler

Default job manager for submitting interactive session jobs. Each session will be scheduled as a single job.

This behaviour can be overridden by each service itself by using `--jobmanager <jobmanager>` option of *interactive.submit*

Note
Your EF Portal installation requires the related grid middleware plugin to be installed and configured. Interactive Plugin will use it to submit and manage interactive session jobs.

Example: `INTERACTIVE_DEFAULT_JOBMANAGER=lsf`

INTERACTIVE_DEFAULT_REMOTE

- value: *optional*
- default: *dcv2*

Default visualization middleware to use.

Available values:

- `dcv2` - use DCV visualization middleware

Example: `INTERACTIVE_DEFAULT_REMOTE=dcv2`

INTERACTIVE_DEFAULT_VNC_QUEUE

- value: *optional*
- default: *(not set)*

Sets the default resource manager queue to use. Interactive session jobs will be submitted on that queue.

This behaviour can be overridden by each service itself by using `--queue <queue name>` option of *interactive.submit*

Example: `INTERACTIVE_DEFAULT_VNC_QUEUE=int_windows`

Limits

INTERACTIVE_DEFAULT_MAX_SESSIONS

- value: *optional*
- default: *undefined (no limits)*

The maximum number of interactive sessions per interactive class.

If you set this default limit to X, each user will be able to start up to X sessions of the same interactive class.

For more information about interactive classes and sessions limits, please refer to Session_limits

Example: `INTERACTIVE_DEFAULT_MAX_SESSIONS=3`

interactive.<remote>.resolutions.conf

Inside this file you can specify some presets of the Remote Visualization Technology desktop geometry as a four-valued colon-separated string plus a label. The label must separated by the previous fields by one or more spaces `widthxheight:fullscreen:allmonitors` label width and height are integers and express the size in pixels, `fullscreen` and `allmonitors` are case-sensitive boolean flags {`true|false`} and label is a human readable string describing the preset.

You can use the keyword auto to let the system guess the current screen resolution. If the list of presets includes a line containing the string custom (no other content on the same line), the user will be able to specify a custom resolution.

Note
Flag allmonitors is meaningful only when fullscreen is true. If you set `allmonitors=true` while `fullscreen=false`, then allmonitors parameter will be automatically converted to false.

Default content of the file:

```
auto Fullscreen on single monitor (autodetect resolution)
5120x1600:true:true Fullscreen on two 30' monitors (5120x1600)
3840x1200:true:true Fullscreen on two 24' monitors (3840x1200)
2560x1600:true:false Fullscreen on single 30' monitor (2560x1600)
1920x1200:true:false Fullscreen on single 24' monitor (1920x1200)
1024x768:false:false Window-mode on single XGA monitor (1024x768)
custom
```

authorization.xconf

This file contains the ACL (Access Control List) definitions specific to Interactive Plugin. It defines some ACLs that are used in the demo portal to allow or deny access to the different visualization middlewares to different users.

For more details on EF Portal ACL system, general EF Portal authorization and its configuration, please refer to EF Portal Administrator Guide, *Security* section, *Authorization System* chapter.

nat.conf

If you set up NAT (Network Address Translation) so that the client machines connect to the cluster nodes through a different IP:PORT pair, this file allows mapping IP:PORT pairs for services running on a node to the corresponding public IP:PORT pair.

Note
Some visualization middleware clients require that the actual port of the service equals the NATted port.

The syntax consists of a line made of two pairs: the real IP:PORT pair followed by the public IP:PORT pair. It is possible to specify a group of ports using the fromPORT-toPORT syntax.

Example:

```
node01 mycompany.com
node01 10.100.0.101
node12:42976 mycompany.com:42976
node01:7900-7910 mycompany.com:5900-5910
node05:7900-7910 10.100.0.101:5900-5910
```

A session starting on host node01, port 7901 would be returned to the client as

```
mycompany.com:5901
```

proxy.conf

If you give access to cluster nodes through a proxy, you can configure this file to assign for each connection a specific proxy server to use.

Note
This configuration applies only to DCV connections.

The default configuration is to have a direct connection from any client to any server, so no proxy for all connections.

The syntax consists of a table, each line has the following columns: PRIORITY, CLIENT-FILTER, SERVER-FILTER, PROXY-TYPE, PROXY-ADDRESS:

PRIORITY

a number to rank the proxy list, 0 is the highest priority

CLIENT-FILTER

the range of IP addresses in the format: NETWORK/PREFIX

Examples:

```
10.20.0.0/16
```

```
0.0.0.0/0 (matches any IP address)
```

SERVER-FILTER

a glob pattern matching the server hostname

Examples:

```
node* (matches any node starting with "node")
node0[1-9] (matches hosts from node01 to node09)
```

PROXY TYPE

the proxy type to use, can be:

HTTP

proxy must support HTTP Connect protocol

SOCKS

proxy must support SOCKS5 protocol

DIRECT

special value to specify no proxy

PROXY-ADDRESS

the proxy hostname and port in the format *host:port* (not used in case proxy type is DIRECT).

Examples:

```
squidproxy.domain:3128
danteproxy:80
10.20.1.1:3128
```

In case multiple proxies with the same priority match, one of them is selected using an internal strategy. In case no proxy for a priority matches, the proxies in the next priority are checked. In case no proxy line matches, an error is returned to the client.

Example: no connection will receive a proxy configuration

```
99 0.0.0.0/0 * DIRECT
```

Example: only connections to node01 will pass through proxyserver:3128, all other connections will be direct.

```
1 0.0.0.0/0 node01 SOCKS proxyserver:3128
99 0.0.0.0/0 * DIRECT
```

Example: connections from IP 10.20.3.20 to node01 will pass through proxyserver:80, other connections to node01 will pass through proxyserver:3128, all other connections will get an error.

```
0 10.20.3.20/32 node01 HTTP proxyserver:80
1 0.0.0.0/0 node01 SOCKS proxyserver:3128
```

url.mapping.conf

The URL mapping configuration allows EF Portal administrators to configure the target endpoints that will be used by clients to connect to the DCV (since 2021.0) remote servers. The configuration file to define the target DCV servers URLs endpoints is `${EF_CONF_ROOT}/plugins/interactive/ url.mapping.conf`.

In this configuration file the administrator can write multiple mappings each one defining a matching rule and a target endpoint. Each rule can match one or more DCV servers as provided upstream by the system by using a set of predefined variables and glob expression. For each match the configuration provides a mapped endpoint that is a triple that includes the host, port and web URL path that will be used by clients to connect to the target DCV server.

Inside the `${EF_CONF_ROOT}/plugins/interactive/url.mapping.conf` configuration file it is possible to use the usual set of EF Portal environment variables available during a service execution (e.g. `EF_*`, session variables) together with the interactive session metadata and a new set of noteworthy variables:

- `${server_host}` - the remote DCV server host as provided by the system in the upstream process;
- `${server_port}` - the remote DCV server port as configured on the DCV server node;
- `${server_web_url_path}` - the DCV server web URL path as configured on the DCV server node;
- `${session_id}` - the DCV session ID;
- `${nat_server_host}` - the value of the DCV server host coming from `${EF_CONF_ROOT}/plugins/interactive/nat.conf;`
- `${nat_server_port}` - the value of the DCV server port coming from `${EF_CONF_ROOT}/plugins/interactive/nat.conf;`
- `${proxy_host}` - the proxy host coming from `${EF_CONF_ROOT}/plugins/interactive/ proxy.conf;`
- `${proxy_port}` - the proxy port coming from `${EF_CONF_ROOT}/plugins/interactive/ proxy.conf;`

Every single value of the tuple, target host, target port and target web URL path, is evaluated separately. Variables are expanded and command substitution executed.

Important
Parameter evaluation is performed on behalf of the user running the Apache Tomcat® server (e.g. efnobody), on the host where EF Portal runs.

Note
The only supported protocol for the mapped URL is HTTPS and cannot be changed.

For further information and examples consult directly the `${EF_CONF_ROOT}/plugins/interactive/url.mapping.conf` configuration file.

xstartup files

The configuration directory also contains a collection of sample xstartup files named `*.xstartup` that may be used in your service definitions to start a X session with the specified Window Manager.

Common tasks for xstartup scripts are, e.g. launching *dbus* daemon, opening an xterm window or setting specific Window Manager parameters.

An example xstartup script:

```
#!/bin/bash
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
xsetroot -solid grey
vncconfig -iconic &
xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
```

The xstartup files shipped with Interactive Plugin are:

- `gnome.xstartup`, xstartup script for GNOME window manager.
- `icewm.xstartup`, xstartup script for ICE window manager.
- `kde.xstartup`, xstartup script for KDE window manager.
- `mate.xstartup`, xstartup script for MATE window manager.
- `metacity.xstartup`, xstartup script for Metacity window manager.
- `mwm.xstartup`, xstartup script for Motif window manager.
- `xfce.xstartup`, xstartup script for Xfce window manager.
- `xfwm4.xstartup`, xstartup script for Xfwm4 window manager.

Default Xstartup are *gnome.xstartup* for desktop sessions and *mwm.xstartup* for standalone interactive applications.

mime-types.xml

This file defines some mime-types useful for Interactive Plugin. Mime-types in this context are used to associate client viewers like DCV® Client to files generated by Interactive Plugin with specific extensions.

File extensions specified in this file are *.dcv*.

For more details on EF Portal mime-types configuration and customization, refer to EF Portal Administrator Guide, *Administration* section, *Common Administration Tasks* chapter.

Interactive Session Life-cycle Extension Points

Interactive Session Dynamic Hooks

EF Portal, starting from version 2017.2, adds two new extension points (hooks) to the interactive session life cycle. In EF Portal version 2025.1 a Pre-Submit Hook was added.

The first customizable hook (`INTERACTIVE_SESSION_PRE_SUBMIT_HOOK`) is executed before the session is started. With a pre-submit hook e.g. an instance or VM can be instantiated or a server can be powered on. Pre-submit hooks are not blocking: they are asynchronously executed by a trigger when the session is in Pending state. If the pre-submit hook script fails, the session is not submitted and goes in Failed state.

For all hooks, the configured hook script is executed by the user running the EF Portal Server. The scripts can be put in any location, as long as it is readable by the Server. Hook script logs are written into the session spooler and are available for reading in the "Details" -> "Session Debug Info" panel for the corresponding session.

Environment variables can be shared between hook scripts to enable execution correlation. Any exported variable which starts with the `SESSION_` prefix is automatically passed along hook script execution, thanks to session spooler metadata.

EF Portal 2025.2 introduced support for propagating additional requirements to the scheduler or DCV Session Manager by defining an environment variable called `SESSION_INTERACTIVE_ADDITIONAL_SUBMITOPTS` in the pre-submit hook. Within this hook, for example an instance can be launched and the relevant information passed along to ensure the session runs on that specific instance.

The second customizable hook (`INTERACTIVE_SESSION_STARTING_HOOK`), typically a shell script, it's called when the session has been successfully set up on the remote host and it's ready to pass to the “Running” state. This hook is meant to execute some simple setup operations at session startup, e.g. to dynamically configure a gateway technology as the AWS Application Load Balancer (ALB) or an Nginx instance, enabling the clients to access the underlying dynamic infrastructure.

The hook has also the capability to add custom metadata to the session and to configure the target host, port and web URL path tuple to be used by the clients to connect to the session.

In order to set the connection parameters to be used by the clients to connect to the interactive session, the hook script has to export the following variables in the environment:

- `INTERACTIVE_SESSION_TARGET_HOST`
- `INTERACTIVE_SESSION_TARGET_PORT`
- `INTERACTIVE_SESSION_TARGET_WEBURLPATH`

These variables will be set as session metadata and used to forge the session URL and connection .dcv file upon a client request through the EF Portal portal. The configuration of these interactive session settings will have the precedence over the static configuration files (e.g. `nat.conf`, `url.mapping.conf`) in defining the connection parameter for the clients.

Inside the hook script, it is possible to use the usual EF Portal environment variables together with the session metadata variables. Noteworthy variables that can be useful to the hook logic are:

- `${INTERACTIVE_SESSION_REMOTE_SESSION_ID}` - the ID for a DCV session;
- `${INTERACTIVE_SESSION_EXECUTION_HOST}` - the execution host of the DCV session as determined internally by the system;
- `${INTERACTIVE_SESSION_DCV2_WEBURLPATH}` - the web URL path of the DCV server, as configured in /etc/dcv/dcv.conf on the DCV server node;
- `${INTERACTIVE_DEFAULT_DCV2_WEB_PORT}` - the web port of the DCV server, as configured in `${EF_CONF_ROOT}/plugins/interactive/interactive.efconf`;

The second customizable hook is executed when the session goes in the “Closed” or “Failed” state.

The starting and closing hooks are configured in the configuration files placed at `${EF_CONF_ROOT}/plugins/interactive/interactive.efconf` (or `${EF_ROOT}/plugins/interactive/conf/interactive.efconf`) through the variables

- `INTERACTIVE_SESSION_PRE_SUBMIT_HOOK`
- `INTERACTIVE_SESSION_STARTING_HOOK`
- `INTERACTIVE_SESSION_CLOSING_HOOK`

respectively.

Sample scripts for pre-submit and closing hooks can be found in `$EF_ROOT/plugins/interactive/bin/samples`.

E.g. `sample.session.pre.submit.hook.sh` and `sample.session.closing.hook.sh`.

Hooks execution is done by the user running the Apache Tomcat® server (e.g. efnobody), and their standard output and standard error are logged in two log files in the interactive session spooler. They are accessible via web from the session details view.

Important
In case of errors the starting hook will block the session startup avoiding the session to go in the “Running” state. If the starting hook fails, it will keep the session in the “Starting” status, and the session will be flagged with a warning message.

The result of the closing hook instead doesn't prevent the session from going into the terminal state. If the closing hook fails the session will anyway terminate and it will be flagged with a warning message.

Warning
The execution of the hooks can be triggered either by a user action (e.g. submission or closing operation) or by the EF Portal internal process that updates the interactive sessions status. At the moment there is no mutual-exclusion mechanism in place and hooks may run concurrently on the same session. It's up to the hook scripts to be concurrency-safe.

Hooks also allow setting custom metadata to the interactive session. Any environment exported variable with prefix `SESSION_` will be added as metadata to the interactive session. All the session metadata are available to the hook scripts.

Sample Starting and Closing Hooks to Configure an AWS ALB

Sample starting and closing hooks to dynamically configure the AWS Application Load Balancer (ALB) on session creation and session closing, are provided in:

- `${EF_ROOT}/plugins/interactive/bin/samples/sample.alb.session.starting.hook.sh`
- `${EF_ROOT}/plugins/interactive/bin/samples/sample.alb.session.closing.hook.sh`

It is suggested to copy the scripts under `${EF_DATA_ROOT}/plugins/interactive/bin` before configuring or modifying them.

These scripts configure an AWS ALB to enable a connection to a host where a DCV interactive session is running.

The starting hook script creates a new Target Group containing the instance where the Session is running and adds a new Listener Rule for the HTTPS listener of the ALB.

The Listener Rule has the role to associate the input URL path to the Target Group. This path must be the web URL path of the DCV server running on the execution node.

Important
Since it is not possible to do URL path translations with an ALB, every DCV server must have an unique web URL path configured. It is suggested to use the hostname of the node as a web URL path for the DCV server running on that node.

The maximum number of Listener Rule(s) per ALB is 100, hence a single ALB can handle at most 100 interactive sessions running concurrently. To increase this limit, please consider adding more ALBs in the infrastructure and to implement a rotation in the starting hook script.

Prerequisites for using the sample hook scripts provided:

- On EF Portal node:
  - AWS Command Line Interface (AWS CLI) must be installed;
  - Since this script is going to be executed by the user running the EF Portal Server, i.e. the Apache Tomcat® user, an AWS CLI profile must be configured for that user, having the permissions to list instances and to manage load balancers. (see CLI Getting Started). Alternatively, if EF Portal is installed in an EC2 instance, a valid AWS role to perform the above mentioned operations should be added to this instance;
  - On AWS account:
    - An AWS Application Load Balancer with an HTTPS listener with a Default Target Group must be already configured and running.

- On DCV server nodes:
  - Each DCV server node must be configured with a unique web URL path (see `/etc/dcv/dcv.conf` configuration file).

The following is an example of the steps to do in order to use the sample AWS ALB hook scripts provided:

- Copy the samples from `${EF_ROOT}/plugins/interactive/bin/samples` to `${EF_DATA_ROOT}/plugins/interactive/bin` and be sure that they are executable.
- Add the two configuration variables inside `${EF_CONF_ROOT}/plugins/interactive/ interactive.efconf`:
  - `INTERACTIVE_SESSION_STARTING_HOOK=${EF_DATA_ROOT}/plugins/interactive/bin/ sample.alb.session.starting.hook.sh`
  - `INTERACTIVE_SESSION_CLOSING_HOOK=${EF_DATA_ROOT}/plugins/interactive/bin/ sample.alb.session.closing.hook.sh`
- Modify the hooks to change the value of the AWS ALB Public DNS name, through the variable `ALB_PUBLIC_DNS_NAME`.
- Configure the AWS role to let the EC2 instance where EF Portal is running to manage the ALB. From the AWS EC2 Console, select EC2 instance -> Instance Settings -> Attach/Replace IAM Role. The following is just an example, more restrictive rules can be used instead:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "elasticloadbalancing:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Common errors from the hook execution:

- "`An error occurred (AccessDenied) when calling the <xyz> operation: User: <abc> is not authorized to perform <xyz>`". The user running the hook (i.e. the user running the Apache Tomcat® server) is not authorized to perform the required operation. The AWS CLI profile must be configured for that user, having the permissions to list instances and to manage load balancers (see CLI Getting Started) or alternatively, if EF Portal is installed in an EC2 instance, configure the correct AWS role for that instance.
- Getting "`502 Bad Gateway`" when connecting to the interactive session. It often means the Target Group of the ALB listener rule is not yet initialized with the target instance. In this case the system usually requires a few more instants before establishing the connection with the instance.

Session limits

Number of sessions

The number of interactive sessions can be limited according to:

- the *interactive session class*. An interactive session class is a group of interactive services. Classes defined and customizable by EF Portal administrators. Interactive classes are defined by setting the following metadata:
    - `INTERACTIVE_CLASS` - an unique identifier for the class.
    - `INTERACTIVE_CLASS_LABEL` - a label for the class (optional).

  The maximum number of interactive sessions for each class is defined by setting the INTERACTIVE_MAX_SESSIONS parameter. The default value is defined in the *interactive.efconf* configuration file.
- the *interactive service*. Each interactive service can be assigned to a certain class and can define the INTERACTIVE_MAX_SESSIONS parameter within its code.
- the *user* or *user group*. With the use of EF Portal ACLs (Access control lists) combined with the first two items. For detailed documentation about ACLs, please refer to the EF Portal Administrator Guide.

A full example follows:

```
<ef:service id="interactive.xterm">
 <ef:name>XTerm</ef:name>
 <ef:metadata attribute="INTERACTIVE_CLASS">xterm</ef:metadata>
 <ef:metadata attribute="INTERACTIVE_CLASS_LABEL">Xterm</ef:metadata>
<ef:metadata attribute="INTERACTIVE_MAX_SESSIONS">3</ef:metadata>
 <ef:option id="project" label="Project"
type="text">Interactive</ef:option>
 <ef:option id="jobmanager" label="Job Manager" type="list">
   <ef:embed id="grid.plugins"/>
 </ef:option>
 <ef:action id="submit" label="Start" result="text/xml">
 "${EF_ROOT}/plugins/interactive/bin/interactive.submit" \
 --name "XTerm" \
 --os "linux" \
 --jobmanager "${jobmanager}" \
 --project "${project}" \
 --remote "dcv" \
 --dcv-xstartup "${EF_ROOT}/plugins/interactive/conf/metacity.xstartup"
\   --close-on-exit \
 --command "xterm"
 </ef:action>
</ef:service>
```

In the example above, the maximum number of sessions of the "xterm" class will be limited to 3 for each user

Note
Interactive session name and interactive class are not dependent on each other.

Log files

`${EF_LOGDIR}/ interactive.log` is the main interactive log file located under the EF Portal log directory.

All other log files (session, debug, authentication) can be found inside each interactive session spooler, and are available via the portal user interface, by reaching the *session details* page.

Interactive Plugin Directory Structure

This section describes the directory structure. Please refer to the EF Portal Administrator Guide for details about the formats and the purpose of the files.

Interactive Plugin is installed in `${EF_ROOT}/plugins/interactive`.

These are the most important contents of the folder:

interactive/

```
|-- WEBAPP
|-- bin
|-- conf
| |-- mappers
| | |-- interactive.duplicated.sessions.xconf
| | `-- interactive.list.sessions.xconf
| |-- dcv2.gpu.balancer.conf
| |-- gnome.xstartup
| |-- icewm.xstartup
| |-- interactive.efconf
| |-- interactive.dcv2.resolutions.conf
| |-- kde.xstartup
| |-- log.xconf
| |-- lxde.xstartup
| |-- mate.xstartup
| |-- metacity.xstartup
| |-- mime-types.xml
| |-- minimal.xstartup
| |-- mwm.xstartup
| |-- nat.conf
| |-- proxy.conf
| |-- template-dcv2.dcv
| |-- xfce.xstartup
| `-- xfwm4.xstartup
|-- etc
|-- lib
`-- tools
```

Interactive Plugin follows the conventional EF Portal plug-in structure and in particular the following directories contain Interactive Plugin system files:

- `WEBAPP` - top level service definition files.
- `bin` - scripts and executables.
- `conf` - configuration files.
- `etc` - Interactive Plugin metadata and EF Portal descriptor files.
- `lib` - internal files used by Interactive Plugin: XML support services and XSL files.
- `tools` - Interactive Plugin integration and interface tools.

Each resource manager plugin supported by Interactive Plugin, includes an *interactive* subdirectory which contains the related interface code with Interactive Plugin:

```
interactive/
|-- interactive.close
|-- interactive.is.session.ready
|-- interactive.log.data
|-- interactive.retrieve.screenshot
|-- interactive.submit
|-- linux.jobscript.functions
`-- services
 `-- interactive.lsf.linux.xml
```

in particular:

- `interactive.submit` - the session submission script.
- `interactive.close` - the operations to be performed to close the session. • services - various service definitions.

Virtual Desktop Administration

Virtual Desktop is implemented by the VDI plugin that defines all the services that interact with the backend to provide the high level, user functionalities.

VDI plugin provides a front-end portal that gives EF Portal administrators an easy way to create, publish and manage Interactive services. End-users instead are provided with a portal to easily access the company Interactive services.

This section explains the configuration files and settings of the VDI plugin.

Topics

Configuration Files

Most of the time the values defined during the VDI plugin installation provide all the information necessary to have a working setup. However the administrator may have the need to change the folders where services files are stored or to change other settings of the system.

The `$EF_TOP/<VERSION>/enginframe/plugins/vdi/conf` subdirectory contains all the VDI plugin configuration files.

Important
As for the other EF Portal plugins, the correct way to change the default configuration is to copy the target configuration file under the `$EF_TOP/conf`, to the `$EF_TOP/conf/plugins/vdi` directory, if it doesn't already exist, and edit the copied file.

Note
Configuration parameters are automatically loaded upon saving. No need to restart EF Portal or logout.

Topics

vdi.conf

This file contains VDI main default configuration parameters.

Users Access Parameters

VDI_ALLOW_ALL_USERS

- value: *required*
- default: *true*

Enables Virtual Desktop access to all the users able to log into the system. If true, the users will be added to the VDI default group at login time.

Available values:

- true - all users are allowed to access to the Virtual Desktop, i.e. VDI plugin services
- false - Virtual Desktop users should be explicitly added or imported by a Views administrator

Example: `VDI_ALLOW_ALL_USERS=true`

service-manager.efconf

This file contains VDI plugin configuration parameters useful for service management.

General Parameters

VDI_SERVICES_ROOT

- value: *required*
- default: `${EF_DATA_ROOT}/plugins/vdi/services` where `EF_DATA_ROOT` is `$EF_TOP/data`

Sets the root folder where services files are stored. Changing this value, you have to change also

- `sdftree` URL value inside href attribute of xi:include tags declared in *vdi.xml, vdi.admin.xml* XML files.
- `load-conf` value of ef:action tags in the services XML files

Example: `VDI_SERVICES_ROOT=${EF_DATA_ROOT}/plugins/vdi/services`

SM_TEMPLATES_ROOT

- value: *required*
- default: `${EF_ROOT}/plugins/vdi/templates`

Sets the root folder where template files for service creation are stored.

Example: `SM_TEMPLATES_ROOT=${EF_ROOT}/plugins/vdi/templates`

Interactive Services Parameters

SM_CATALOG_INTERACTIVE

- value: *required*
- default: *${VDI_SERVICES_ROOT}/catalog*

Sets the folder where unpublished interactive services files are stored.

Example: `SM_CATALOG_INTERACTIVE=${VDI_SERVICES_ROOT}/catalog`

SM_PUBLISHED

- value: *required*
- default: *${VDI_SERVICES_ROOT}/published*

Sets the folder where published interactive services files are stored.

Example: `SM_PUBLISHED=${VDI_SERVICES_ROOT}/published`

interactive.editor.efconf

This file contains configuration parameters for the interactive service editor in the Virtual Desktop.

Note
In EnginFrame version 2015.0 this file was named `vdi.editor.efconf` and had a slightly different set of configuration parameters.
During the installation of a newer version, EF Portal makes a copy of the old configuration file under `$EF_TOP/conf/plugins/vdi` directory, naming it as vdi.editor.efconf.backup.

Interactive Editor Parameters

VDI_EDITOR_OS

- value: *optional*
- default: *windows,linux*

Sets supported operating systems for interactive sessions. Comma separated list without any blank space.

Available values:

- windows - Windows® Desktop
- linux - Linux® Desktop

Example: `VDI_EDITOR_OS=windows,linux`

VDI_EDITOR_CLUSTERS

- value: *optional*
- default: Cluster ids retrieved from system

Sets cluster ids to list on service editor. Comma separated list without any spaces. Example:

```
VDI_EDITOR_CLUSTERS=clusterid1,clusterid2
```

VDI_EDITOR_CLUSTERS_ARCH_clusterId

- value: *optional*
- default: linux for all the clusters id

Sets supported operating systems for interactive sessions scheduled in a specific cluster. Comma separated list without any spaces. clusterId is one of the cluster id defined in the `VDI_EDITOR_CLUSTERS` list.

Available values:

- windows - Windows® Desktop
- linux - Linux® Desktop

Example:

```
VDI_EDITOR_CLUSTERS_ARCH_lsfCluster=linux
VDI_EDITOR_CLUSTERS_ARCH_myCluster=windows,linux
```

VDI_EDITOR_REMOTES

- value: *optional*
- default: *dcv2*

Sets the list of supported remote visualization technologies to display in the service editor. Comma separated list without any spaces.

Available values:

- dcv2 - DCV

Example: `VDI_EDITOR_REMOTES=dcv2`

VDI_EDITOR_REMOTES_ARCH_remoteId

- value: *optional*
- default: linux, windows for dcv2

Sets supported session types for a specific remote id. Comma separated list without any spaces. remoteId is one of the remote id defined in the `VDI_EDITOR_REMOTES` list.

Available values:

- windows - Windows® Desktop
- linux - Linux® Desktop
- linux-app - Linux® Desktop application

Example: `VDI_EDITOR_REMOTES_ARCH_dcv2=linux`

VDI_EDITOR_DESKTOP_MANAGERS

- value: *optional*
- default: *none*

Sets the list of supported desktop manager ids to display on the service editor. Comma separated list without any spaces.

For each desktop manager, the name to display could also be specified in the configuration parameter `VDI_EDITOR_DESKTOP_MANAGER_NAME_`*`desktopManagerId`*. If omitted it will be equal to the id.

For each desktop manager, the path to the xstartup file must be specified in the configuration parameter `VDI_EDITOR_DESKTOP_MANAGER_XSTARTUP_`*`desktopManagerId`*. Interactive plugin already provides a set of preconfigured xstartup files for supported desktop managers under `${EF_ROOT}/plugins/interactive/conf/` directory.

Example:

```
VDI_EDITOR_DESKTOP_MANAGERS=gnome,kde


VDI_EDITOR_DESKTOP_MANAGER_NAME_gnome=GNOME
VDI_EDITOR_DESKTOP_MANAGER_XSTARTUP_gnome=${EF_ROOT}/path_to_gnome_xsta
rtup
VDI_EDITOR_DESKTOP_MANAGER_XSTARTUP_kde=/path_to_kde_xstartup
```

Log files

The main VDI log file is located under the EF Portal log directory, `${EF_LOGDIR}/vdi.log`, where `EF_LOGDIR` is `$EF_TOP/logs/<hostname>`.
Log files can be reviewed using the “View Log Files” service available in the Troubleshooting section of the Operational Dashboard portal.

VDI Plugin Directory Structure

This section describes the directory structure.

VDI plugin is installed in `${EF_ROOT}/plugins/vdi`.

These are the most important contents of the folder:

```
vdi/
|-- WEBAPP
|-- bin
|-- conf
| |-- authorization.xconf
| |-- log.xconf
| |-- service-manager.efconf
| |-- interactive.editor.efconf
| `-- vdi.conf
|-- etc
|-- lib
`-- templates
```

VDI follows the conventional EF Portal plug-in structure and in particular the following directories contain VDI system files:

- `WEBAPP` - top level service definition files and web resources.
- `bin` - scripts and executables.
- `conf` - configuration files.
- `etc` - VDI plugin metadata and EF Portal descriptor files.
- `lib` - internal files used by VDI plugin: XML support services and XSL files.
- `templates` - Interactive services templates.

Interactive services are installed in `$EF_TOP/data/plugins/vdi/services`.

These are the important contents of the folder:

```
services/
|-- catalog
|-- published
```

```
`-- extra
```

The following directories contain VDI plugin services files:

- `catalog` - root folder for unpublished services
- `published` - root folder for published services
- `extra` - root folder for custom extra services to be included in the Virtual Desktop.

HPC Workspace Administration

HPC Workspace is implemented by the Applications plugin that defines all the services that interact with the backend to provide the high level, user functionalities.

HPC Workspace provides a front-end portal that gives EF Portal administrators an easy way to create, publish and manage batch and interactive services. End-users instead are provided with a portal to easily access the company HPC services.

This section explains the configuration files and settings of the HPC Workspace / Applications plugin.

Important
The service examples provided by the Workspace make use of a `JOB_WORKING_DIR` and assume it is mounted by both EF Portal hosts and execution hosts.
By default the `JOB_WORKING_DIR` is set to the `EF_SPOOLER` directory of the submitted service, so in order to use the examples the root spoolers directory should be shared with the execution hosts.
This is not a requirement of EF Portal Workspace but a simplification used by the examples.

Topics

- Configuration files
- Log files
- Applications Directory Structure

Configuration files

Most of the time the values of the settings collected during the Applications plugin installation provide all the information necessary to have a working setup. However, the administrator may have the need to change the folders where services files are stored or to change other settings of the system.

All the Applications Plugin configuration files are located in the `$EF_TOP/<VERSION>/enginframe/ plugins/applications/conf` subdirectory.

Important
As for the other EF Portal plugins the correct way to change default configuration is to copy the target configuration file under the `$EF_TOP/conf`, to the `$EF_TOP/conf/plugins/applications` directory, if it doesn't already exist, and edit the copied file.

Topics

- applications.conf
- service-manager.efconf
- interactive.editor.efconf

Note
Configuration parameters are automatically loaded upon saving. No need to restart EF Portal or logout.

applications.conf

This file contains the main default configuration parameters for Applications.

Users Access Parameters

APPLICATIONS_ALLOW_ALL_USERS

- value: *required*
- default: *true*

Enables HPC Workspace access to all the users able to log into the system. If true, the users will be added to the Applications default group at login time.

Available values:

- true - If this value is set, all users are allowed to access the HPC Workspace.
- false - If this value is set, workspace users can be explicitly added or imported by an HPC Workspace administrator.

Example: `APPLICATIONS_ALLOW_ALL_USERS=true`

service-manager.efconf

This file contains Applications plugin configuration parameters that you can use for service management.

General Parameters

APPLICATIONS_SERVICES_ROOT

- value: *required*
- default: `${EF_DATA_ROOT}/plugins/applications/services` where `EF_DATA_ROOT` is `$EF_TOP/data`

Sets the root folder where services files are stored. A change to this value requires that you also make changes to the following:

- `sdftree` URL value inside href attribute of `xi:include` tags declared in *applications.xml, applications.admin.xml* XML files
- `load-conf` value of `ef:action` tags in the services XML files

Example:

```
APPLICATIONS_SERVICES_ROOT=${EF_DATA_ROOT}/plugins/applications/service
```

`s` SM_TEMPLATES_ROOT

- value: *required*
- default: *${EF_ROOT}/plugins/applications/templates*

Sets the root folder where service templates files are stored.

Example: `SM_TEMPLATES_ROOT=${EF_ROOT}/plugins/applications/templates`

SM_PUBLISHED

- value: *required*
- default: *${APPLICATIONS_SERVICES_ROOT}/published*

Sets the root folder where Applications stores files for published services.

Example: `SM_PUBLISHED=${APPLICATIONS_SERVICES_ROOT}/published`

Batch Services Parameters

SM_CATALOG_BATCH

- value: *required*
- default: *${APPLICATIONS_SERVICES_ROOT}/catalog/batch*

Sets the folder where unpublished batch services files are stored.

Example: `SM_CATALOG_BATCH=${APPLICATIONS_SERVICES_ROOT}/catalog/batch`

Interactive Services Parameters

SM_CATALOG_INTERACTIVE

- value: *required*
- default: *${APPLICATIONS_SERVICES_ROOT}/catalog/interactive*

Sets the folder where unpublished interactive services files are stored.

Example:

`SM_CATALOG_INTERACTIVE=${APPLICATIONS_SERVICES_ROOT}/catalog/interactiv`

`e` interactive.editor.efconf

This file contains configuration parameters for the interactive service editor in the Workspace.

The file syntax and parameters are the same as in the interactive.editor.efconf in the *Virtual Desktop Administration* VDI plugin section.

Log files

The main Applications log file is located under EF Portal log directory: `${EF_LOGDIR}/applications.log`, where `EF_LOGDIR` is `$EF_TOP/logs/<hostname>`. Log files can be reviewed using the “View Log Files” service available in the Troubleshooting section of the Operational Dashboard portal.

Applications Directory Structure

This section describes the directory structure.

Applications are installed at this location: `${EF_ROOT}/plugins/applications`. The following

are the most important contents of the folder:

```
applications/
|-- WEBAPP
|-- bin
|-- conf
| |-- authorization.xconf
| |-- log.xconf
| |-- service-manager.efconf
| |-- interactive.editor.efconf
| `-- applications.conf
|-- etc
|-- lib
`-- templates
```

Applications follows the conventional EF Portal plug-in structure and, in particular, the following directories contain Applications system files:

- `WEBAPP` - top-level service definition files and web resources.
- `bin` - scripts and executables.
- `conf` - configuration files.
- `etc` - Applications plugin metadata and EF Portal descriptor files.
- `lib` - internal files that are used by the Applications plugin. They are XML support services and XSL files.
- `templates` - Services templates.

By default, both batch and interactive services are installed at this location: $EF_TOP/data/plugins/applications/services.

The following are the most important contents of the folder:

```
services/
|-- catalog
| |-- batch
| `-- interactive
|-- published
`-- extra
```

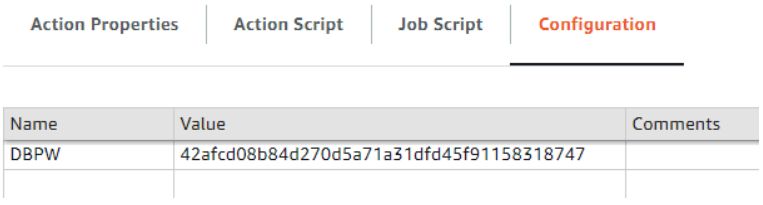
The following directories contain Applications services files:

- `catalog` - root folder for both batch and interactive unpublished services.
- `published` - root folder for published services.
- `extra` - root folder for custom extra services to be included in the HPC Workspace portal.

Job History Service

The Job History service is available in the Application Services Admin section and can be enabled there.

The Job History creates new tables to store long term job history data in the Derby DB. DB Triggers will be installed the first time the service is run. To access the DB the Derby DB password needs to be configured in the service. Please edit the “Job History” service and click on “Submit”. In the “Configuration” please configure the DB password which you can find in `$EF_CONF_ROOT/derby/derby.properties` as value of “`derby.user.dbadmin`”. E.g.



Then “Close”, “Save” and “Close” the service and “Publish” the service. You can also “Test Run” to verify the Job History service functionality. The service also offers a demo mode to display sample job history data.

Managing spoolers

This chapter provides an overview of the concepts that are related to EF Portal spoolers and their management.

A spooler is a dedicated data container that EF Portal creates to host files that users provide (for example, input files uploaded using a web browser) or files that are generated by the services (for example, output or temporary files).

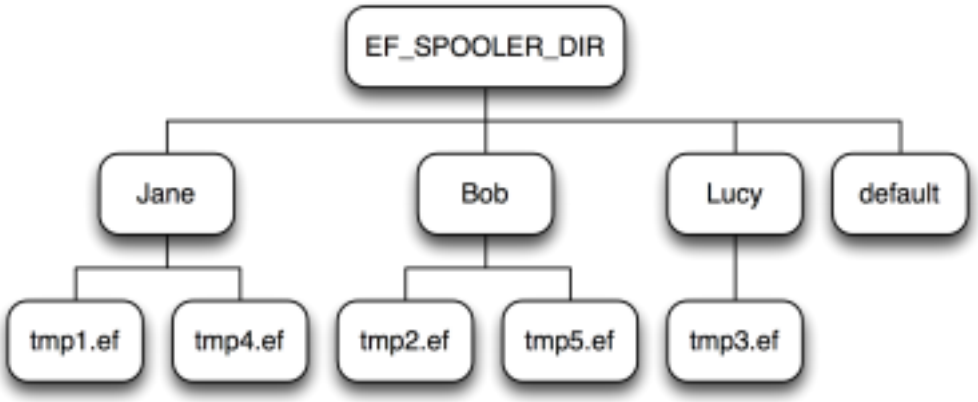
Every time the service is run (unless explicitly configured) causes EF Portal to create a new

spooler with appropriate user permissions that allow services to read from and write to the spooler's directory.

Under `EF_SPOOLER_DIR`, EF Portal creates a directory for each user the first time that the system is accessed. These directories are named using the user's names. Under each <username> directory, EF Portal creates its spoolers, one for each time the service is run.

The following image provides a visual overview of EF Portal's spooler directory structure.

Spoolers directory structure:



In the preceding image, Jane, Bob, Lucy each have their own spooler directories. tmp1.ef, tmp2.ef, tmp3.ef, tmp4.ef, tmp5.ef are spooler directory names that EF Portal created dynamically.

Spoolers requirements

If agents that use the spoolers run on hosts other than the server hosts, the spoolers must reside on a *shared file system*. This shared file system must be mounted from the EF Portal Server's host. Both the agents and server must be able to read and write to the shared file system. With theEF Portal's mapping mechanism, these file systems can be mounted with different paths on server and agent hosts.

Note
If you're using multiple agents, the same mount point path must be used for all of them.

`EF_SPOOLER_DIR` directory must be owned by the user that's running EF Portal Server (for example, efnobody). The same user must have read and write permissions. This is because EF Portal Server initially creates spoolers for users. All other users must also have read and run permissions for the spooler root directory. In other words, the spooler root directory must have the following ownerships and permissions: `rwx r-x r-x efnobody:efnobody` where efnobody is efnobody's primary group.

The spoolers area must be placed on a file system where the agent's owner has complete read and write privileges. Depending on who runs agent daemon, this can be achieved by one of the following tactics:

- Avoiding *root squashing* on the NFS server when the agent owner is *root*.
- Using the same user running the server when the agent owner is a *normal user*.

Note
Distinct EF Portal deployments can't share the same spooler area.

Spooler security permissions

EF Portal Server starts with a user file creation mask, such as **umask**, set to 022. This means the following happens:

- Files are created with 644 `rw-r--r--` permissions.
- Directories are created with 755 `rwxr-xr-x` permissions.

This assures the user running the EF Portal Server can both read and write files in spoolers. This might lead to a weak security environment because every user can read files from spoolers that are owned by other user accounts. For this reason, EF Portal Agent changes spooler permissions to 750 `rwxr-x---` just before the service is run. In other words, the final spooler permissions are read, write, and run for the spooler's owner and read and run for EF Portal Server's owner. This occurs because user efnobody with group efnobody creates the spooler. EF Portal Agent changes spooler's owner by running chown <username> <spooler> and spooler ownership becomes <username>:efnobody. This allows the spooler's owner to perform whatever action is needed inside the spooler and that only efnobody members are allowed to read data inside the spoolers. A security best practice is to only have efnobody as a member of the efnobody group.

Configuring EF Portal spoolers

This section describes how to customize basic spoolers settings concerning directory paths and file download aspects.

Topics

- Configuring spoolers default root directory
- Download files from spoolers

Configuring spoolers default root directory

`EF_SPOOLER_DIR` directory is chosen during installation. All spoolers are created by EF Portal under this directory. The default value is `$EF_TOP/spoolers`.

Spoolers root directory can be changed by editing *`EF_SPOOLERDIR`* parameter inside `$EF_TOP/conf/enginframe.conf`. The startup parameter name `EF_SPOOLERDIR` is slightly different from the property EF_SPOOLER_DIR that's set and used inside EF Portal.

Example Change the spooler default root directory

```
EF_SPOOLERDIR=/mnt/scratch/spoolers
```

After this change, all spoolers are created under `/mnt/scratch/spoolers` directory.

Note

Changes to this parameter require an EF Portal Server restart.

Note

Spooler location is defined at *creation time*, so changing `EF_SPOOLERDIR` doesn't affect an existing spooler's location. EF Portal Agent retrieves old spoolers from the old location and new spoolers from the new location.

Refer to Spoolers requirements when setting up a new spooler root directory.

Download files from spoolers

Files are downloaded from spoolers on the user's behalf through the server and agent. EF Portal Server receives a download request and forwards it to the agent that accesses the file. The agent posts the file back to the server using HTTP or HTTPS.

This process implies EF Portal Agent is able to connect back to the server using HTTP or HTTPS for sending data. As a result, make sure that you are careful with how you configure your network and firewall.

Note
If EF Portal Server is configured to accept requests through *HTTP over SSL* - HTTPS - protocol, then refer to the Configuring HTTPS topic to configure the server and agent.

Consider setting a *mime-type* for downloaded files to let browsers identify the file's type. You can customize and configure the mime-type that EF Portal uses when downloading files. Managing internet media types provides instructions on how to set up mime-type configurations.

Configure download URL on agent

The download process highlights how an agent has to connect back to the server for sending downloaded data. Usually the server HTTP endpoint, that is, the host and port that the agent connects to, is automatically detected from the server's request.

There might be network configurations or architectural scenarios that require specific configurations. For example, web access might be configured with an HTTP server in front of EF Portal Server. In this case, the agent must use a different host and port to connect to the server. To enable this, you must explicitly configure the complete URL that the agent must use to connect to the EF Portal Server.

The `ef.download.server.url` parameter inside `$EF_TOP/conf/enginframe/agent.conf` (or `$EF_TOP/<VERSION>/enginframe/conf/agent.conf`) sets the URL agent uses to connect to server. It's defined as follows:

```
ef.download.server.url=http[s]://<host>:<port>/<web-context>/download
```

Replace the following elements with your specific details:

- The host and port value identifies the EF Portal Server network endpoint.
- The web-context value is the root context that you choose when installing EF Portal. By default, it's EF Portal.

The following is modified with example values for illustration purposes.

```
ef.download.server.url=http://localhost:8080/enginframe/download
```

Configure streaming download timeout

Streaming downloads have a timeout setting. If the file being streamed doesn't change during this set interval of time, EF Portal interrupts its stream that's being sent to a client. By default, this value is set to 300 seconds.

This value is specified in `$EF_TOP/conf/enginframe/server.conf` (or `$EF_TOP/<VERSION>/enginframe/conf/server.conf`). You change this value by editing `ef.download.stream.inactivity.timeout` property inside

`$EF_TOP/conf/enginframe/ server.conf`. The value is expressed in seconds.

In the following example, the timeout value is set to 600 seconds (10 minutes).

```
ef.download.stream.inactivity.timeout=600
```

Configure streaming download sleep time

The file streaming download feature of EF Portal works using a pull model. The server periodically queries an agent for available data.

You can set the specific interval of time between two subsequent checks. By default, it's set to 5 seconds. A lower interval time might improve the user experience but it might also increase system load.

This value is specified in `$EF_TOP/conf/enginframe/server.conf` (or `$EF_TOP/<VERSION>/enginframe/conf/server.conf`). You change this value by editing `ef.download.stream.sleep.time` parameter. The value is expressed in seconds.

For example, with `ef.download.stream.sleep.time=20`, the sleep time is set at 20 seconds.

Spooler life cycle

This section outlines EF Portal's spooler lifecycle. Besides outlining the spooler's life cycle, each sub section describes common customizations that you can apply to EF Portal.

Overview

Spoolers are created for each service submission. More specifically, spoolers are created for all those services whose spooler definition has a TTL different from -1.

Spoolers are initially created by EF Portal Server with a defined time-to-live. Together with the spooler directory the server also creates an entry in EF Portal's spooler database called *repository*. The repository is file-system based and each entry is a file that contains all information that's necessary to recreate a spooler.

- The owner
- The physical location path on both server and agent
- The display name
- And other properties

The server also has the responsibility to save user's files into a spooler before contacting an agent for running a service. After the spooler setup tasks have been accomplished, the server contacts an agent to run the service. The agent first changes the spooler's and its contents ownership and then uses this spooler as the working directory for the service it runs.

EF Portal removes a spooler when its life-time expires, deleting the spooler's directory with its content and its repository entry. An EF Portal thread called *reaper* periodically checks if there are expired spoolers ready to be removed.

Change repository location

The default EF Portal repository path is `$EF_TOP/repository`. With EF Portal, you can change the location of where repository entries are stored.

For example, you might want to save repository entries on a high speed and reliable file-system or

on an area that has "dynamic" file-system paths (for example, a directory with contents that change often) to conform to your company's policies.

The repository location is configured by the `EF_REPOSITORYDIR` parameter inside the `$EF_TOP/conf/` enginframe.conf file. It specifies an absolute path in the file system. You can use other variables that are defined in efportal.conf for path definition. For example, `EF_REPOSITORYDIR=$EF_TOP/` repository defines a repository location using the `$EF_TOP` variable. `EF_REPOSITORYDIR=/mnt/nas/ef` repository sets an absolute path for repository directory.

Note
If you make changes to `EF_REPOSITORYDIR`, you must restart EF Portal Server.

Configure reaper sleep time

The EF Portal reaper is a thread that periodically wakes up to check if there are expired spoolers in the system needing cleanup.

You can configure the reaper's thread sleep interval to specify how frequently this check is performed.

This value is specified in `$EF_TOP/conf/enginframe/server.conf` (or `$EF_TOP/<VERSION>/ enginframe/conf/server.conf`). You change this value by editing the `ef.reaper.sleep.time` property. The value is expressed in minutes. By default, it's set to 30 minutes.

In this example, `ef.reaper.sleep.time=60`, the sleep time is set to 60 minutes (1 hour) between each thread's reap.

Spoolers removal: Dead spoolers

If for any reason spooler cleanup fails, spooler's directory is renamed with the `DEAD_` prefix added in front of its original name.

For example, if EF Portal can't remove spooler `tmp32062.ef`, it's renamed to `DEAD_tmp32062.ef`. You can remove dead spoolers from your system safely.

If you set up a deadspooler logging target after a dead spooler was created, the event is logged to the `$EF_TOP/logs/DEAD_spoolers.{agent|server}.log` depending on whether the server or agent had an error.

Custom Places: Additional File and Remote Spooler Places in User File View

NI SP EF Portal introduces easy creation of additional Places to allow for quick access to local or remote file locations frequently visited.

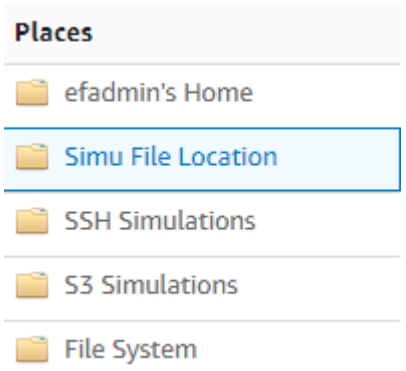
In the example, below, the file locations **Simu File Location**, **SSH Simulation Files** and **S3 Simulation Files** are created under Places by putting the following code into `~/.ef/ef.places` (permission 600):

A list of locations with URL and description separated by ~ (URLs should not include blanks):

```
file:///home/simu/simu_3_💯/simu_results~Simu File Location
```

```
ssh://simu@simu.ni-sp.com:/home/simu~SSH Simulations
s3://@simu-bucket/~S3 Simulations
```

Will create Places similar to this (in case of the admin the general filesystem is added as well):



There are two options to configure EF Places. As a user and in the global configuration for every user. There is also a configuration of allowed places to control the places the users can configure.

Administrators can define File Manager places for all users using either (in ascending priority order):

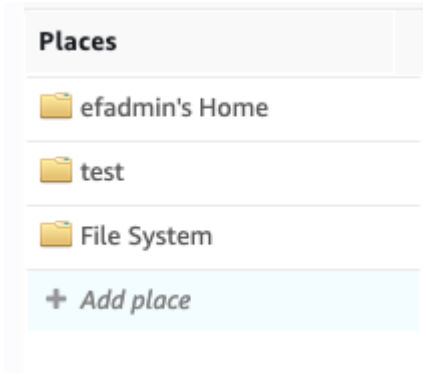
- `$EF_CONF_ROOT/plugins/fm/ef.places`
- `$EF_ROOT/plugins/fm/conf/ef.places`

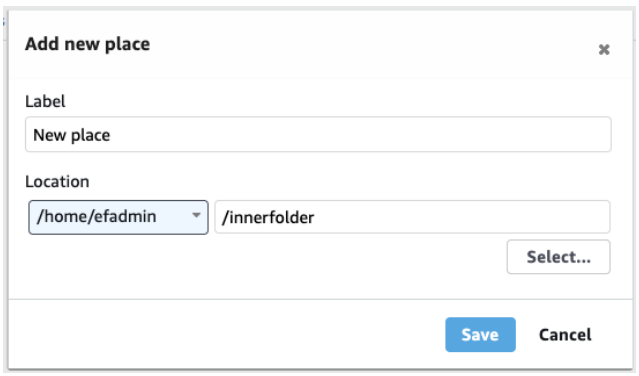
Moreover, user defined places in `$HOME/.ef/ef.places` which point to directories outside their home directory, must be explicitly allowlisted by administrators to be used. Allowlisting is performed using either (in ascending priority order):

- `$EF_CONF_ROOT/plugins/fm/ef.places.allowed.conf`
- `$EF_ROOT/plugins/fm/conf/ef.places.allowed.conf`

`$HOME` directory is allowed by default. Places defined by administrators for all users are allowed by default as well.

Starting from EF Portal 2026.0, it is possible to configure allowed places and add or edit existing *Places* directly through the user interface.





Places created through the user interface are stored in the `~/.ef/ef.places` file.
If you are using a version prior to 2026.0, or if you prefer to modify the file manually, you can still edit it directly.

For the s3 remote spooler to work the aws cli needs to be installed and configured with the proper authentication to enable the aws s3 commands to work on the respective s3 bucket.

Support for Copy/Paste/Move operations in Spoolers and Places

When selecting files or directories the user is offered to copy or paste/move the files or directories. The user can move to a different spooler, remote spooler or place and copy or paste/move the files or directories into the present directory. Filenames should not include special characters ‘ “ < > &.

To enable this functionality the json parser “`jq`” should be installed. If it is not installed the user will also be informed by a message that jq should be installed. When copying or pasting it will not be checked if the target files are already existing.

Prevent File Downloads by Optional Allowlist

To prevent download, preview or streaming of files via EF Portal and allow only specific users in case to download files a new feature and configuration files have been introduced:

- `$EF_ROOT/plugins/fm/conf/fm.download.allowed.conf`
- `$EF_CONF_ROOT/plugins/fm/fm.download.allowed.conf`

```
# Every line is a user or a regex matching a set of user
# Comments with # are supported as part of the configuration
#efadmin
#ef.*
#root
```

Managing the sessions directory

This chapter illustrates the basic concepts that concern EF Portal interactive session data and its management.

The data of an interactive session is in a dedicated data container. This container is created by EF Portal to host the files that are required for the interactive session lifecycle. These include the thumbnail of a screenshot, for example.

`INTERACTIVE_SHARED_ROOT` is organized in the same way as the EF Portal spoolers. For more information, see Managing spoolers. Under `INTERACTIVE_SHARED_ROOT`, EF Portal creates a directory for each user the first time an interactive session is created. These directories are named with the user's names. Under each <username> directory, EF Portal creates its session data directory, one for each interactive session.

Sessions requirements

Sessions must reside on a *shared file system* that must be mounted from the EF Portal Server's host, EF Portal Agent's host and visualization nodes. This area might not require to be shared when submitting sessions to some Distributed Resource Managers. For more information, see Shared file system requirements.

The `INTERACTIVE_SHARED_ROOT` directory must be owned by the user that's running EF Portal Server (for example, efnobody) and by efnobody's primary group. It must have the 3777 permissions. To summarize, the following permissions are required:

`d rwx rws rwt efnobody:efnobody` where efnobody is efnobody's primary group.

The EF Portal installer creates this directory with the proper permissions on your behalf. The

default location is `$EF_TOP/sessions`. You can change the `INTERACTIVE_SHARED_ROOT` value in the `$EF_TOP/conf/plugins/interactive/interactive.efconf` file.

This shared area has to be readable and writable by EF Portal nodes and visualization nodes. EF Portal provides a mapping mechanism that enables these file-systems to be mounted with different paths on EF Portal and visualization hosts. This configuration is specific to the Distributed Resource Manager that's used for the session and can be configured in the specific plugin configuration file:

- On PBS Professional®, use `PBS_INTERACTIVE_SHARED_ROOT_EXEC_HOST` in the `$EF_TOP/conf/ plugins/pbs/ef.pbs.conf` file.
- On SLURM™, use `SLURM_INTERACTIVE_SHARED_ROOT_EXEC_HOST` in the `$EF_TOP/conf/ plugins/slurm/ef.slurm.conf` file.
- On SGE, use `SGE_INTERACTIVE_SHARED_ROOT_EXEC_HOST` in the `$EF_TOP/conf/plugins/sge/ ef.sge.conf` file.

Managing Host View

Configurable Quick Commands for Host Information

To access information on Linux cluster hosts quickly, EF Portal 2025.0 introduced quick commands available in the upper right of the respective host information. This is supported for Linux schedulers and access to the hosts can be configured via ssh or a scheduler command.

Using either one of the following conf file:

- `$EF_ROOT/plugins/hydrogen/conf/host.commands.conf (preinstalled)`
- `$EF_CONF_ROOT/plugins/hydrogen/host.commands.conf`

admins can define one or more global actions to run custom commands on the host the user is viewing, and display the command output in a dialog window. Quick Commands are available to admins only by default.

Administrators can also allow regular users to use and also override admin defined commands, setting the configuration parameter `HY_HOST_INFO_COMMANDS_ADMIN_ONLY` to "false" in either `$EF_ROOT/plugins/hydrogen/conf/host.commands.conf` or `$EF_CONF_ROOT/plugins/hydrogen/host.commands.conf`. In this case, users can define their actions using the file `$HOME/.ef/host.commands.conf`.

New commands can be added with the syntax “`description|command|question`”. A question can be configured for further input to the command which is stored in `%INPUT%` and can then be used in the command replacing the value. Pipes are allowed in the command as well escaped with `\|`.

The remote command to execute can be configured. The default is ssh with examples for SLURM and LSF.

Every admin user - and standard users as well in case allowed - can configure their own commands overwriting the system-wide configured commands in the file `~/.ef/host.commands.conf`.

Example configuration in `host.commands.conf`:

```
# System processes
PS|ps -ef
W|w
# Performance monitoring
Top|top -b -n 1 -w 120 --sort %CPU
Free|free -h
# Storage information
DF|df -h
DF Grep|df -h \| grep %INPUT%|Grep for
# SLURM
#sinfo|sinfo -l
#show hosts|scontrol show nodes
#show node|scontrol show node=%INPUT%|Which host to show?
# Hardware information
#lspci|lspci
#lsusb|lsusb
```

In `ui.hydrogen.conf ($EF_ROOT/plugins/hydrogen/conf/ui.hydrogen.conf` or `$EF_CONF_ROOT/plugins/hydrogen/ui.hydrogen.conf)` the Host Info functionality can be configured:

```
# Host Info
HY_HOST_INFO_SERVICE_URI="//ui.hydrogen/grid.host.info.data"
HY_HOST_INFO_COMMANDS_CONF="host.commands.conf"
HY_HOST_INFO_COMMANDS_ADMIN_ONLY="true"
HY_HOST_INFO_COMMANDS_REMOTE_CMD="ssh" # ssh from EF server as logged
in user
# HY_HOST_INFO_COMMANDS_REMOTE_CMD="srun -w"   # SLURM
# HY_HOST_INFO_COMMANDS_REMOTE_CMD="lsrun -m"  # LSF
```

Limiting the Hosts Displayed in the Hosts View

In certain cases the host list should only show a limited set of hosts. This can be implemented with the new configuration introduced via

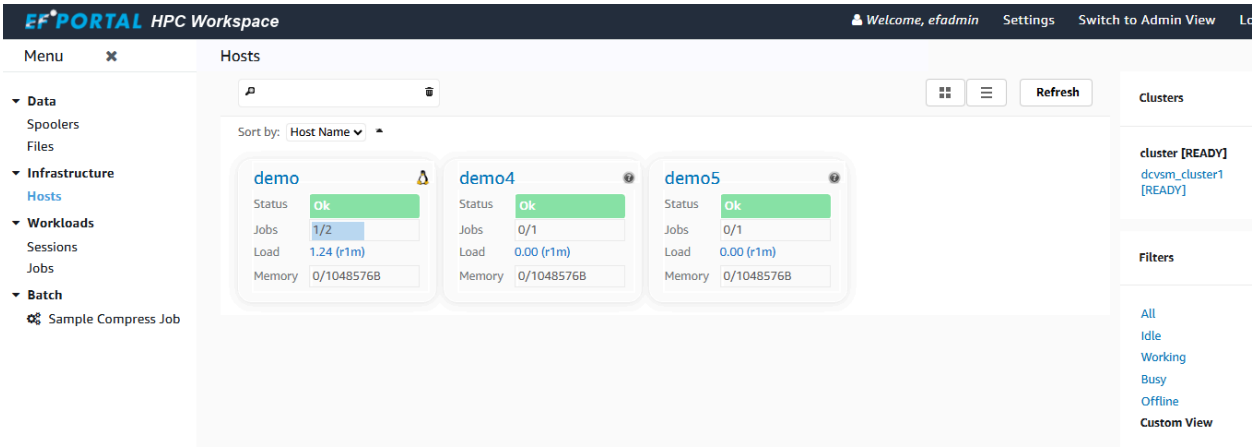
- `$EF_ROOT/plugins/hydrogen/conf/list.hosts.custom.filter.conf`
- `$EF_CONF_ROOT/plugins/hydrogen/conf/list.hosts.custom.filter.conf`

With this configuration only hosts matching certain regexps are shown in the Custom View available in the Filters on the right side:

```
# Here you can configure a custom filter for List Hosts view
# to show only a selected group of hosts based on their names.
# Every line not commented is a host name or a regex matching a set of
host names.
# Examples:
#   node-1
#   node-[3-4]
```

```
#    node-[1-2,6-7].*
demo
demo[4-5]
```

Here is an example with the Custom View active in the right Filter selection:

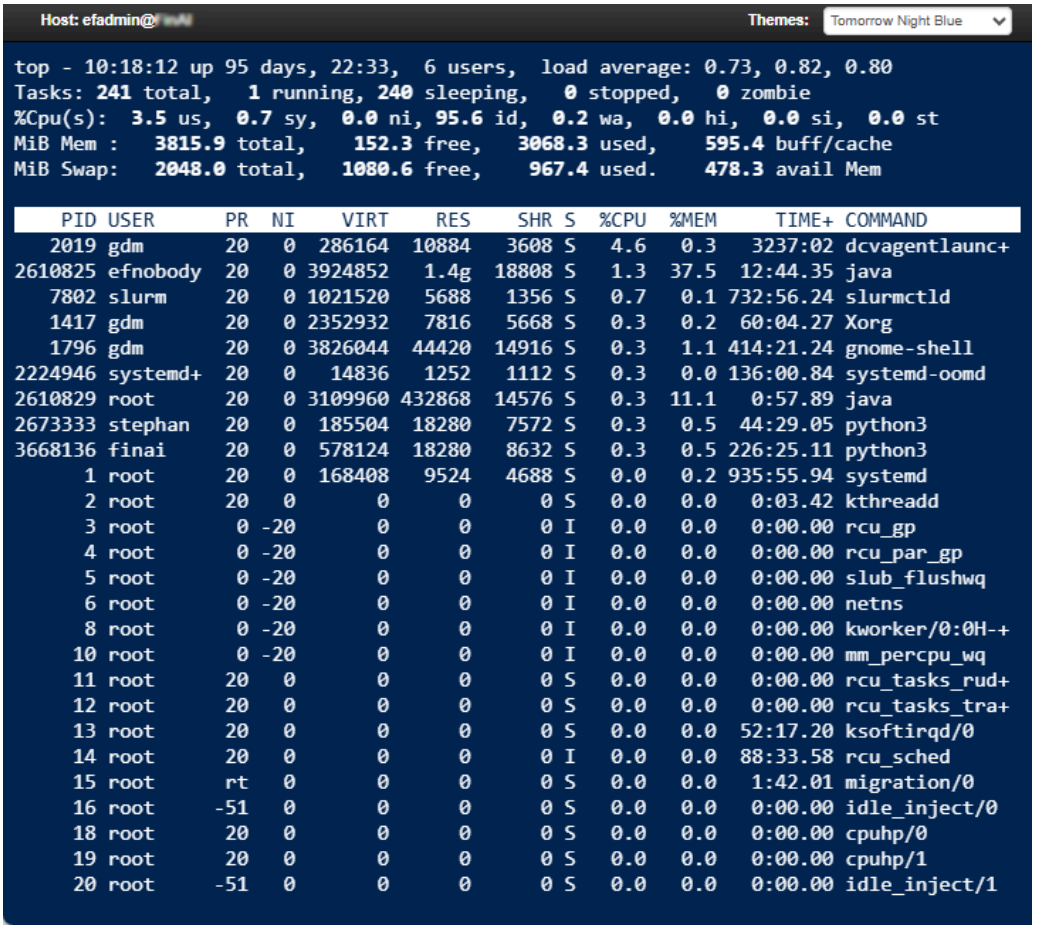


The custom view can be selected as default view with a configuration in `ui.hydrogen.conf`.

Web Terminal

The Web Terminal provides administrators with secure, browser-based access to remote hosts directly from the EF Portal Hosts View. In addition to the quick commands we introduced in EF Portal 2025.0 the new “Terminal” functionality as highlighted in the following screenshot from the EF Portal Hosts View:

When selecting Terminal, a new browser window opens and presents an interactive shell session on the selected host:



The terminal supports multiple visual themes, selectable from the upper-right menu, allowing users to customize the appearance of the session.

Runtime environment

The Web Terminal is powered by Node.js, which is bundled with EF Portal - no external installation is required.

Note: The Web Terminal uses a version of Node.js that requires glibc 2.28 or newer. Distributions with older glibc versions - such as CentOS 7, RHEL 7, and Ubuntu 16.04 - are not supported, and the Web Terminal will be automatically disabled on these systems.

Startup Behavior

The *shell* feature is enabled by default. The enablement is controlled by the `EF_ROLES` or `EF_ADDITIONAL_ROLES` parameters in the `$EF_TOP/conf/enginframe.conf` file.

During EF Portal startup (`systemctl start enginframe`), the startup script verifies that the embedded Node.js runtime is functioning correctly.

If Node.js fails to start - for example, on an unsupported operating system - the script automatically disables the Web Terminal and creates a flag file to signal to EF Portal that the feature is unavailable.

When the Web Terminal is disabled, the Terminal button is automatically hidden from the Host Details page.

If you want to start/stop only the Web Terminal component you can execute:

```
$EF_TOP/bin/enginframe start|stop shell
```

Configuration

Web Terminal configuration can be managed through the Web Configuration Editor (available since EF Portal 2026.0) under *Cluster Access -> Features*, or via the environment file located in: `$EF_TOP/conf/shell`.

Key parameters include:

```
# Duration of inactivity (in milliseconds) after which the remote shell connection will be closed.
# Default: 300,000 milliseconds (5 minutes).
# SHELL_INACTIVE_TIMEOUT_MS=300000

# Maximum allowed duration (in milliseconds) for a remote shell connection.
# Default: 3,600,000 milliseconds (60 minutes).
# SHELL_MAX_DURATION_MS=3600000

# Command used to establish the remote shell connection to the host (e.g., rsh).
# Default: ssh
# SHELL_SSH_WRAPPER=ssh
# SHELL_SSH_WRAPPER="srun --pty --time=60 --nodelist=%NODE% /bin/bash"
# SHELL_SSH_WRAPPER="lsrun -m %NODE% /bin/bash"

# A colon (:) separated list of hostnames that defines the accessible hosts in the cluster.
# Example: host01:host03:host05
# Default: Empty list
# SHELL_SSHHOST_ALLOWLIST=

# Path to your private key file used for securing HTTP connections (HTTPS).
SHELL_PRIVATE_KEY=/opt/nisp/enginframe/conf/node/certs/node.key.pem

# Path to your certificate file used for securing HTTP connections (HTTPS).
SHELL_CERTIFICATE=/opt/nisp/enginframe/conf/node/certs/node.cert.pem

# Port number on which the EF Portal Remote Shell Server listens for incoming connections.
# Default: 3000
SHELL_PORT=3000
```

The remote host can be accessed in the Web Terminal via ssh or interactive scheduler commands like `srun` or `lsrun`.

The logfiles of the Web Terminal can be found in the files `ef.shell.stdout` and `ef.shell.stderr` in the standard EF Portal log directory, eg. `$EF_TOP/logs/$HOSTNAME`.

EF Portal Proxy

The EF Portal Proxy provides a secure, centralized entry point for accessing web-enabled applications running on compute nodes, such as JupyterLab and DCV. Instead of requiring users to connect directly to multiple node ports (e.g., `node1:8443`, `node2:8891`), all traffic is funneled through a single, consistent EF Portal host and port. This greatly simplifies access and improves overall network security.

The Proxy listens on a single `PROXY_PORT` and acts as a reverse proxy, routing incoming HTTP, HTTPS, and WebSocket traffic to the correct backend host and port based on the request path.

Both static and dynamic routing patterns are supported, allowing the proxy to handle fixed endpoints as well as applications running on different nodes and ports.

Typical use cases include exposing JupyterLab and DCV sessions through stable, user-friendly URLs - without opening additional ports on compute nodes or exposing them directly to users.

Instead of accessing services directly on compute nodes (e.g., `node1:8443` or `node2:8891`), users connect to the EF Portal host on `PROXY_PORT`, and the proxy transparently forwards traffic to the correct backend service based on the URL structure.

Runtime environment

The Proxy is powered by Node.js, which is bundled with EF Portal - no external installation is required.
Note: The Proxy uses a version of Node.js that requires glibc 2.28 or newer. Distributions with older glibc versions - such as CentOS 7, RHEL 7, and Ubuntu 16.04 - are not supported, and the Proxy will be automatically disabled on these systems.

Startup Behavior

To enable the proxy feature, set the `EF_ADDITIONAL_ROLES=proxy` parameter in the `$EF_TOP/conf/enginframe.conf` file (or enable it through the Web Configuration Editor) and then start EF Portal with:

```
$EF_TOP/bin/enginframe start
```

or, if you want to start/stop only the Proxy component:

```
$EF_TOP/bin/enginframe start|stop proxy
```

Configuration

Proxy configuration can be managed through the Web Configuration Editor (available since EF Portal 2026.0) under *Cluster Access -> Features*, or via the environment file located in: `$EF_TOP/conf/proxy`.

By default, the Proxy runs over HTTPS on port `3001` using a self-signed certificate.

Both the port and the certificate settings can be configured in the `$EF_TOP/conf/proxy/env` configuration file. This is the port your users' browsers and backend apps (e.g., Jupyter) must target.

```
# Port number on which the EF Portal Proxy Server listens for incoming
connections.
# Default: 3001
PROXY_PORT=3001

# Path to your private key file used for securing HTTP connections
(HTTPS).
PROXY_PRIVATE_KEY=/opt/nisp/enginframe/conf/node/certs/node.key.pem

# Path to your certificate file used for securing HTTP connections
(HTTPS).
PROXY_CERTIFICATE=/opt/nisp/enginframe/conf/node/certs/node.cert.pem
```

Routing rules are defined in `$EF_TOP/conf/proxy/proxy.config.yaml`.
The proxy can route requests statically (fixed host/port) or dynamically (host and port extracted from the URL).

Proxy routing configuration

The primary parameters use regular expressions (RegEx) to match the incoming URL and extract the dynamic backend host and port.

- `path: "RegEx"`

  Defines the pattern that triggers this rule. Must match the public URL structure expected by users.

- `portRegex: "RegEx"`

  Defines the RegEx with a capture group `(...)` to extract the backend port number from the path. Required for dynamic proxy. When it is set, the proxy operates in dynamic mode.

- `hostRegex: "RegEx"`

  Defines the RegEx with a capture group `(...)` to extract the backend hostname from the path. Optional, but recommended, for dynamic proxy.

- `secure: true|false`

  Disables verification for the backend connection (proxy to compute node). Required to accept self-signed certificates from the application. Default: `true`

- `rewrite: "RegEx"`

  Defines which part of the path is removed before forwarding. Required for example for DCV.

- `protocolRewrite: http|https`

When set to `https`, it forces HTTPS to the backend (needed for example for DCV, because WebSocket streaming uses `wss://`). Default: `http`.

- `target: String`

  Specifies the target host and port. It serves as an alternative to `hostRegex` and `portRegex`; when it is set, the proxy operates in static mode.

The logfiles of the Proxy can be found in the files `ef.proxy.stdout` and `ef.proxy.stderr` in the standard EF Portal log directory (eg. `$EF_TOP/logs/$HOSTNAME/ef.proxy.std*`).

Below are examples for Jupyter Notebook, Jupyter Lab and DCV.
A preconfigured Jupyter Notebook service is also available in the EF Portal and works out of the box when the proxy is enabled via the `EF_ADDITIONAL_ROLES` variable in the `enginframe.conf` file. You only need to set the proxy address variable in the Jupyter Notebook service configuration.

Dynamic Jupyter routing

The proxy supports multiple Jupyter instances running simultaneously on different nodes and ports. To reach each session, you can use for example a path of the form: `/jupyter/<hostname>/<port>`

Example of `$EF_TOP/conf/proxy/proxy.config.yaml` configuration:

```
proxies:
  - path: "^/jupyter/[^/]+/[0-9]+"
    hostRegex: "^/jupyter/([^/]+)/[0-9]+"
    portRegex: "^/jupyter/[^/]+/([0-9]+)"
    secure: false
```

This preconfigured proxy path is available in the EF Portal to simplify use with the bundled Jupyter Notebook service.

Configuration parameters meaning

- `path`

  Matches URLs like `/jupyter/node1/8891`.

- `hostRegex`

  Extracts the backend hostname from the URL.
  Example: `/jupyter/node1/8891` → host `node1`.

- `portRegex`

  Extracts the backend port.
  Example: `/jupyter/node1/8891` → port `8891`.

- `secure: false`

Accept self-signed certificates if the Jupyter endpoint is TLS-enabled (or just ignore TLS verification for HTTPS targets).

Jupyter Lab application requirements

To function correctly behind the reverse proxy, the Jupyter **Lab** application must be configured to handle the altered URL structure and WebSocket traffic. These parameters are typically set within the job submission script or when launching the application.

Parameters required for Jupyter Lab 3.2.9 are listed below. See the official Jupyter documentation for additional information and updates.

- `--port=...`

  Backend port where Jupyter listens. Defines the specific backend port that the proxy must target (`portRegex` captures this).

- `--notebook-dir=...`

  Defines the notebook root directory.

- `--ServerApp.base_url=/prefix/...`

  CRITICAL: Ensures the application generates all internal URLs (links, redirects) with the full proxy path prefix so they are correctly routed back through the proxy.

- `--ServerApp.tornado_settings=...`

  Sets `static_url_prefix` so JS/CSS are loaded from the correct path behind the proxy.

- `--ServerApp.websocket_url=wss://...`

  CRITICAL: Directs interactive WebSocket traffic (used for terminals and kernels) back to the public proxy endpoint, bypassing the compute node address.
  This must point at the public proxy endpoint, e.g.
  `wss://PROXY_PUBLIC_ENDPOINT:PROXY_PORT/jupyter/node1/8891`
  You must use `ws://` or `wss://` depending on whether Jupyter is running over HTTP or HTTPS.

- `--ServerApp.allow_origin='*'`

  Allows the browser to connect to the proxied application (CORS mitigation).

- `--ServerApp.allow_remote_access=True`

  Allows clients that are not localhost to connect (needed when accessed via proxy).

Example of Jupyter Lab command

Please replace `USERNAME`, `PROXY_PUBLIC_ENDPOINT`, `PROXY_PORT`, `COMPUTE_NODE` and `COMPUTE_PORT` parameters.

```
jupyter lab --port=COMPUTE_PORT --notebook-dir /home/USERNAME \
  --ServerApp.base_url=/jupyter/COMPUTE_NODE/COMPUTE_PORT/ \
```

```
--ServerApp.tornado_settings="{'static_url_prefix':'/jupyter/COMPUTE_NODE/COMPUTE_PORT/static/'}" \
--ServerApp.websocket_url=wss://PROXY_PUBLIC_ENDPOINT:PROXY_PORT/jupyter/COMPUTE_NODE/COMPUTE_PORT \
  --ServerApp.allow_origin='*' \
  --ServerApp.allow_remote_access=True
```

Users would then access:

```
https://PROXY_PUBLIC_ENDPOINT:PROXY_PORT/jupyter/COMPUTE_NODE/COMPUTE_PORT
```

Jupyter Notebook application requirements

To function correctly behind the reverse proxy, the Jupyter **Notebook** application must be configured to handle the altered URL structure. These parameters are typically set within the job submission script or when launching the application.

Parameters required for Jupyter Notebook 6.4.10 are listed below. See the official Jupyter documentation for additional information and updates.

- `--port=...`

  Backend port where Jupyter listens. Defines the specific backend port that the proxy must target (`portRegex` captures this).

- `--notebook-dir=...`

  Defines the notebook root directory.

- `--NotebookApp.base_url=/prefix/...`

  CRITICAL: Ensures the application generates all internal URLs (links, redirects) with the full proxy path prefix so they are correctly routed back through the proxy.

- `--NotebookApp.tornado_settings=...`

  Sets `static_url_prefix` so JS/CSS are loaded from the correct path behind the proxy.

- `--NotebookApp.allow_origin='*'`

  Allows the browser to connect to the proxied application (CORS mitigation).

- `--NotebookApp.allow_remote_access=True`

  Allows clients that are not localhost to connect (needed when accessed via proxy).

Note that `--NotebookApp.websocket_url` parameter is required for Jupyter Lab, but it is handled automatically and should not be configured for Jupyter Notebook.

Example of Jupyter Notebook command

Please replace `USERNAME`, `PROXY_PUBLIC_ENDPOINT`, `PROXY_PORT`, `COMPUTE_NODE` and `COMPUTE_PORT` parameters.

```
jupyter notebook --port=8893 --no-browser --notebook-dir="$EF_SPOOLER"
\
--NotebookApp.base_url="/jupyter/COMPUTE_NODE/8893/" \
--NotebookApp.tornado_settings="{'static_url_prefix':'/jupyter/COMPUTE
_NODE/8893/static/'}" \
--NotebookApp.allow_origin='*'   --NotebookApp.allow_remote_access=True
\
"${EF_SPOOLER}/${file_ipynb}
```

Users would then access:

```
https://PROXY_PUBLIC_ENDPOINT:PROXY_PORT/jupyter/COMPUTE_NODE/COMPUTE_
PORT
```

Dynamic DCV routing

The proxy supports multiple DCV instances running simultaneously on different nodes and ports. To reach each session, you can use for example a path of the form: `/dcv/<hostname>/<port>`

Example of `$EF_TOP/conf/proxy/proxy.config.yaml` configuration:

```
proxies:
  - path: "^/dcv/[^/]+/[0-9]+"
    hostRegex: "^/dcv/([^/]+)/[0-9]+"
    portRegex: "^/dcv/[^/]+/([0-9]+)"
    rewrite: "^/dcv/[^/]+/[0-9]+"
    protocolRewrite: "https" # required for WSS
    secure: false # To use DCV self-signed certificates
```

Configuration parameters meaning:

- `path`

  Matches URLs like `/dcv/hostname/8444`.

- `hostRegex`

  Extracts the backend hostname from the URL.
  Example: `/dcv/node1/8444` → host `node1`.

- `portRegex`

  Extracts the backend port.
  Example: `/dcv/node1/8444` → port `8444`.

- `rewrite`

Defines which part of the path is removed before forwarding. Typically you forward `/` or the remainder after `/dcv/<host>/<port>`.

- `protocolRewrite: "https"`

  Forces HTTPS to the backend (needed because DCV WebSocket streaming uses `wss://`).

- `secure: false`

  Allows self-signed DCV certificates.

Notification System

The in-portal notification system provides instant, actionable messages with clear severity levels, read/unread tracking, and role-based visibility (RBAC). It supports multi-tenant scoping to isolate tenant data and presents a consistent, accessible UI for all users.

The notification system can be set up to notify end users about scheduler job progress and results. It also issues warnings to administrators for important portal states — such as high memory usage, low storage capacity, or low license availability.

Administrators can also send custom notifications to all users in a role or to selected users via the dedicated interface. This can be done using the **Send Notification** service in both *HPC Workspace (Applications)* and *Virtual Desktops (Views)* administration portals.

Notifications are stored in the database configured for EF Portal

Enabling Browser Desktop Notifications

The notification system supports native browser desktop notifications so you can receive alerts even when you are working in other tabs or applications. To receive these notifications, ensure your browser and operating system both permit notifications for this site. The portal must be accessed over *https://* (browsers do not allow notification permissions on insecure origins).

When EF Portal first requests permission, choose *Allow* in the browser prompt. If you previously dismissed or blocked it, you can re-enable it from the site settings:

- Chrome/Edge: click the padlock icon in the address bar → *Site settings* (or *Permissions*) → set *Notifications* to *Allow*.
- Firefox: click the padlock icon in the address bar → *Permissions* → *Send Notifications* → *Allow*.
- Safari (macOS): open Safari → *Settings* (or *Preferences*) → *Websites* → *Notifications* → set EF Portal site to *Allow*.

Also confirm OS-level notification settings are enabled for your browser:

- Windows: *Settings* → *System* → *Notifications* → enable notifications for your browser; ensure *Focus assist* is off or configured to show notifications from your browser.

- macOS: *System Settings* → *Notifications* → select your browser → enable *Allow Notifications*; choose *Banners* or *Alerts* and enable *Badges/Sounds* if desired; check that *Focus/Do Not Disturb* isn't suppressing alerts.

Configuring Automatic Notifications

Use the Configuration Editor to control which automatic notifications the system sends to users and admins.

Under *HPC* → *Job Notifications*, administrators can enable job-state notifications from the scheduler and choose which available states should trigger delivery. You may select any subset of available states: **Running**, **Done**, **Exit**; by default, none are selected. When a configured state is reached, a notification is sent to the user who owns the job. Apply and save changes in the *Configuration Editor* to activate the settings.

Under *Server* → *Advanced* → *Notifications*, administrators can enable system-health notifications and configure threshold values for monitored resources: **CPU Load**, **Host Storage**, and EF Portal **License Tokens**. These notifications are delivered only to administrators. Thresholds are configurable (default: 95%); setting a threshold to 0 disables notifications for that specific event. In the same section you can also configure the deduplication interval for notifications having the same Category, Role and Deduplication ID. Apply and save changes in the Configuration Editor to activate the settings.

Triggering Notifications via API

Administrators and integrators can trigger the Notification System on demand to send custom notifications, using either the provided Bash CLI utility or the in-page JavaScript utility. These tools are useful for testing, automation, or emitting custom alerts from in-portal workflows.

Please refer to the **Notifications** services in the *Technology Showcase* for practical examples of how to use the provided tools.

You can also create custom notifications via the REST API for out-of-band scripts and workflows.

Access Control

Administrators are permitted to send notifications to any role and any user. Regular users are restricted to sending notifications only to themselves, and only with the `user` role. These controls are enforced server-side and cannot be bypassed by altering client-side parameters; if a request attempts to target an unauthorized audience or role, the server will coerce it to the allowed scope.

The server determines the sender's identity from the authenticated principal. For API calls made by Bash or other automation, the effective user is the account that authenticates the request — either the user running the script when using host-integrated credentials, or the owner of the REST API token presented in the Authorization header. For calls originating from JavaScript within the portal, the effective user is the currently logged-in user associated with the active session; client code cannot impersonate another user. All notification creation requests are audited with the resolved principal and target audience. If broader targeting is required than a regular user's scope permits, use an administrator account or a dedicated service account with the necessary permissions.

Bash CLI Utility

The `send.notification` shell command can be used from **action scripts**, **triggers**, or any shell executed by the EF Portal Server — where `EF_ROOT` is set. It internally calls the `//notification.system/create` service via `<ef.call.service>` tag. It provides a convenient way to send notifications from scripts without needing to know the underlying service invocation details.

Synopsis:

```
send.notification [OPTIONS]

Options:
  -t, --title       Notification title (required)
  -b, --body        Notification body
  -s, --severity    SEVERITY. One of: info, warning, error, critical.
                    Default: info
  -c, --category    CATEGORY. One of:
                    system, security, scheduler, maintenance, other.
                    Default: system
  -r, --role        ROLE. One of: user, admin. Default: admin
  -u, --user        Target username. Omit to broadcast to ROLE.
      --dedup-id    Deduplication ID.
                    Prevents duplicates within 30 mins by default.
                    A duplicate has the same CATEGORY, ROLE and
                    Deduplication ID.
```

JavaScript Utility

The `$.enginframe.sendNotification()` method is a *jQuery* extension provided by the EF Portal framework. It sends a notification by calling the internal *//notification.system/create* service via AJAX and returns a *Promise*. This is useful for building custom portal pages, dashboards, or service forms that need to notify users about events or status changes without requiring a server-side shell script. This is the same API used by the **Send Notification** administrator service.

Synopsis:

```
$.enginframe.sendNotification({
  title:    string, // Notification title (required)
  body:     string, // Notification body
  severity: string, // One of: info, warning, error, critical.
                    // Default: info
  category: string, // One of:
                    // system, security, scheduler, maintenance,
other.
                    // Default: system
  role:     string, // One of: admin, user. Default: admin
  user:     string, // Target username. Omit to broadcast to ROLE
  dedupId:  string  // Deduplication ID.
                    // Prevents duplicates within 30 mins by default.
```

```
                    // A duplicate has the same CATEGORY, ROLE and
                    // Deduplication ID.
});

// Returns a Promise:
//   .then((response) => { message: string, notificationId: string,
//                         dedupId: string })
//   .catch((err) => { status: number, statusText: string,
//                     responseText: string })
```

REST API Service

To send a notification from a workflow that runs outside the EF Portal server — for example, a **job script** executing on a scheduler's compute node — create the notification via the Portal REST API.

Use the following values to call the appropriate service:

- **`sdf`**: either `/applications/applications.xml` or `/vdi/vdi.xml`
- **`service`**: `//notification.system/create`

The service offers the same options shown in the paragraphs above.

Please refer to the EF Portal REST API chapter for more information.

DCV Session Manager

DCV Session Manager is a set of installable software packages that includes an Agent and a Broker. It also includes an API that you can use to build frontend applications that create and manage the lifecycle of DCV sessions across a fleet of DCV servers.

The DCV Session Manager integration is only supported with DCV 2020.2 and later.

Node monitoring, Linux console sessions, autorun files, and session screenshots are new features that were introduced in DCV 2021.0. These features have been added in EF Portal 2020.1 and are only supported by DCV 2021.0 and later.

For more information, see What is DCV Session Manager in the *DCV Session Manager Administrator Guide*.

How to set up DCV Session Manager

When you install NI SP EF Portal, you can choose to use DCV Session Manager to manage DCV sessions.

To allow EF Portal to interact with DCV Session Manager, register EF Portal as a Session Manager API client.

If you don't have a client ID and client password for EF Portal, you must request these credentials from your DCV Session Manager Broker administrator. For more information about registering your client API with the Broker and obtaining a client ID and password, see register-api-client in the *DCV Administrator Guide*.

To properly configure the DCV Session Manager, the installer requires the following information:

- The base URL of the remote DCV Session Manager. This URL is used by EF Portal to manage DCV sessions.
- The full URL, client ID, and client password that EF Portal uses to authenticate with the Broker.

The installer stores this information in the following locations:

- `$EF_CONF_ROOT/plugins/dcvsm/clusters.props`

  This file contains sensitive information that's used to authenticate to remote DCV Session Managers. Because of the potential for abuse, this file must have strict permissions. We recommend that you only grant read and write permissions for the user that is running the Apache Tomcat® server. Do not make this file accessible to other users.

  ```
  -rw------- efnobody efnobody
  ```

- `$EF_CONF_ROOT/plugins/dcvsm/dcvsm.efconf`

  This is the configuration file. It contains the configuration for the DCV Session Manager plugin integration and includes sensitive information.

The following is an example `clusters.props` file.

```
# Configuration for cluster dcvsm_cl1

# The OAuth2 Client ID is assigned by the DCV Session Manager Broker
# when registering EF Portal as an API client.
# This value uniquely identifies EF Portal to the broker and is required
# to get an access token.
# To retrieve both the Client ID and the Client Password,
# run the following command on the broker host:
#    dcv-session-manager-broker register-api-client --client-name EF
# Example: DCVSM_CLUSTER_dcvsm_cluster1_AUTH_ID=abcd1234
DCVSM_CLUSTER_dcvsm_cl1_AUTH_ID=clientID

# The OAuth2 Client Password associated with the Client ID above.
# This password is required to get an OAuth2 access token from the broker.
# Treat this value as a secret and ensure file permissions on clusters.props
# restrict access appropriately.
# This value is returned together with the Client ID when running:
#    dcv-session-manager-broker register-api-client --client-name EF
# Example: DCVSM_CLUSTER_dcvsm_cluster1_AUTH_PASSWORD=xyz987
DCVSM_CLUSTER_dcvsm_cl1_AUTH_PASSWORD=clientPassword

# The URL used by EF Portal to request OAuth2 access tokens.
# This must point to the broker's /oauth2/token endpoint.
# Example: https://broker-hostname:8448/oauth2/token
DCVSM_CLUSTER_dcvsm_cl1_AUTH_ENDPOINT=https://sm-hostname:sm-port/oauth2/token

# Base URL of the DCV Session Manager Broker API.
# EF Portal uses this endpoint to call APIs such as describeServers,
# createSession, listSessions, etc.
# Example: https://broker-hostname:8448
DCVSM_CLUSTER_dcvsm_cl1_SESSION_MANAGER_ENDPOINT=https://sm-hostname:sm-port

# Whether EF Portal should skip TLS certificate validation when connecting
# to the broker.
# Set to:
# - true – disable certificate verification (useful for self-signed certs)
# - false – enforce strict TLS validation (recommended for production)
# Default: false (if not specified)
```

```
DCVSM_CLUSTER_dcvsm_cl1_NO_STRICT_TLS=false
```

The following is an example of the dcvsm.efconf file.

```
# DCV Session Manager - EF Portal Configuration
#
# This file defines which DCV Session Manager clusters are available
# to EF Portal.
# Each cluster listed in DCVSM_CLUSTER_IDS must have a matching
# configuration block in: $EF_TOP/conf/plugins/dcvsm/clusters.props
#
# - Cluster IDs must be comma-separated (no spaces).
# - Each cluster ID must match exactly the CLUSTER_ID used in the variable
#   names inside clusters.props.
# - If a cluster ID is listed here but missing in clusters.props, EF Portal
#   will not be able to communicate with that DCV Session Manager instance.
#
# List of DCV Session Manager cluster identifiers managed by EF Portal.
# Example: DCVSM_CLUSTER_IDS=dcvsm_cluster1,dcvsm_cluster2
DCVSM_CLUSTER_IDS=dcvsm_cl1
```

The name of the cluster that's reported in `DCVSM_CLUSTER_IDS` must match the name that's used for the properties that are described in `clusters.props`. If this isn't the case, EF Portal can't retrieve the cluster configuration parameters.

The installer also automatically configures the following files:

- `$EF_CONF_ROOT/plugins/interactive/interactive.efconf`

  Contains `EF_DELEGATE_SESSION_MANAGERS=dcvsm`.

  This enables customization of the default connection behavior from the User's Settings view.
- `$EF_CONF_ROOT/plugins/grid/grid.conf`

  Configures `dcvsm` as grid manager. This allows DCV SM Hosts to be displayed on the *Hosts* page within the HPC Workspace or Virtual Desktop view.

Add more DCV Session Manager clusters

To add more clusters, you can use the EF Portal installer. It creates DCV Session Manager clusters that use a default configuration. You can change these settings after the cluster is created.

For each new cluster, the following is required:

- A new clusterId must be added in `DCVSM_CLUSTER_IDS` (in a comma separated list) in `$EF_CONF_ROOT/plugins/dcvsm/dcvsm.efconf`.
- A new set of parameters must be added in `$EF_CONF_ROOT/plugins/dcvsm/clusters.props`.

Enable hosts monitoring for DCV Session Manager

By default, DCV Session Manager hosts are not shown in the *Hosts* section of the portal. To enable this feature, edit two configuration files to meet the following requirements:

- The `dcvsm` string must be added in `EF_GRID_MANAGERS` (comma separated list) in `$EF_CONF_ROOT/ plugins/grid/grid.conf`.
- The `dcvsm` string must be added in `GRID_XML_PLUGINS` (comma separated list) in `$EF_ROOT/ plugins/grid/conf/jobcache.efconf`.

After you change these properties, you must restart EF Portal to apply the changes.

Configuring DCV Session Manager secure connection

By default, the installer configures a secure connection using strict TLS checking between EF Portal and the DCV Session Manager Broker. We recommend that you use this default configuration.

If you so choose, you can disable this behavior by setting `DCVSM_CLUSTER_clusterID_NO_STRICT_TLS` to true in the clusters.props configuration file. If a valid certificate is presented, EF Portal trusts the DCV Session Manager Broker.

If the DCV Session Manager Broker is configured to use self-signed certificates, make sure that the root certificate is added to the Tomcat JVM KeyStore.

For instructions on how to add a root certificate to the Tomcat JVM KeyStore, see Configuring HTTPS.

In the following example, the `CA_ROOT.pem` file that's provided by the DCV Session Manager Broker is used.

```
# 1. export the Session Manager CA Certificate in der format with
openssl openssl x509 -in CA_ROOT.pem -inform pem -out ca.der -outform
der

# 2. Verify that Java Keytool is able to read the certificate
keytool -v -printcert -file ca.der

# 3. import Session Manager CA Certificate in JVM keytool
keytool -importcert -alias dcvsm \
 -keystore $JAVA_HOME/lib/security/cacerts \
 -storepass java_keystore_password \
 -file ca.der

# 4. verify that the root certificate has been imported
keytool -keystore "$JAVA_HOME/lib/security/cacerts" \
 -storepass java_keystore_password \
 -list | grep dcvsm
```

How to create a new remote desktop interactive service

An Admin user can create a new remote interactive service using a Linux scheduler or DCV Session Manager from the Virtual Desktop admin portal.

DCV Session Manager cluster supports both Windows and Linux hosts.

Session modes

The Interactive plugin with DCV Session Manager, creates VIRTUAL or CONSOLE sessions in Linux hosts and CONSOLE sessions in Windows hosts. For more information about session types and support, see DCV Session Manager documentation.

VIRTUAL sessions support one or more sessions per instance. This is the default for Linux hosts.

Console sessions support only one session per instance. This is the default for Windows hosts.

Note: VIRTUAL and CONSOLE sessions can't be active on the same host at the same time.

Create a remote desktop interactive service

Create and publish a remote desktop interactive service

1. Open the EF Portal portal and **Login** as Admin user.
2. Choose the **Virtual Desktop**.
3. Choose **Switch to Admin View** in the header navigation pane.
4. In the sidebar navigation pane, choose **Manage** and then **Interactive Services**.
5. Choose **New** to create an interactive service.
6. Select **Create from Template** and choose **Create**.
    a. Choose **Settings**. Here you can customize your service. Choose **Close** to skip to the next step.
    b. In the yellow box, type in a new name for your service, such as "MyService".
    c. Choose **Launch Session**.
    d. Review and keep the settings, including the **Session Mode** in the **Execution Environment** section.
    e. Choose **Close**, **Save**, and **Close**.
7. To modify or test the service, select your new service and choose **Edit**.
8. To publish the service, select your new service and choose **Publish**, then select **All Users** and **Publish**.
9. Choose **Switch to User View** in the header navigation pane.
10. Your new remote interactive service is displayed under **Services** and available for use.

Using DCV Session Manager, you can tag the host where the DCV Server is running with custom values. You can also use these values to select which host the session is created on. For more information, see the DCV Session Manager documentation.

With EF Portal session placement, you can create an interactive service that targets hosts with specific characteristics. Using this feature, you don't need to recall specific details such as the hostname or IP address and delegate their discovery to the remote DCV Session Manager Broker.

To specify a set of "tags" for targeting the host, add the submitopts option in the ActionScript tab:

```
vdi.launch.session --submitopts "ram_gb='4' Software='My Software' server:Hostname='myhostname'"
```

- Multiple tags can be specified, separated by spaces.
- Each tag is a key-value pair that's in the format of "`key=value`". There can't be any spaces before and after the equal sign (=) unless the keys and values are wrapped in single quotes (').

DCV options support for Linux scheduler VDI submit

Following Linux scheduler VDI submit options are available as options to the `vdi.launch-session` command in the Linux Desktop action-script:

- `--dcv2-permissions-file`
- `--dcv2-client-eviction-policy`
- `--dcv2-disable-login-monitor`
- `-dcv2-gl`
- `--dcv2-storage-root (e.g. --dcv2-storage-root %home%/Desktop)`
- `--dcv2-init`

- `--dcv2-max-concurrent-clients`
- `-dcv2-pid-check-ttl (controls the behavior of the check for the Xdcv process)`

If no hosts are available, the session fails to be created. This can happen if there are no hosts or if all the hosts are in use and aren't available to create the session.

Starting from EF Portal 2020.1 there is the possibility to specify a custom Autorun File to launch when a session is created. The file must be present on the remote host within a reserved folder. For more information about autorun files, see DCV Session Manager documentation.

This feature is only compatible with DCV Session Manager 2021.0+ on CONSOLE sessions on Windows DCV servers and VIRTUAL sessions on Linux DCV servers. It is not supported with CONSOLE sessions on Linux DCV servers.

To specify the Autorun File, add the delegate-autorun-file option in the ActionScript tab. You can do this by running the following command.

```
vdi.launch.session --delegate-autorun-file
"autorun/relative/file/path.sh"
```

This parameter can also have arguments via the option delegate-autorun-arguments separated by ",". Arguments containing spaces do not need to be surrounded with escaped double quotes, moreover this approach is discouraged. No character needs escaping except for the comma itself. E.g.

```
vdi.launch.session --delegate-autorun-file
"autorun/relative/file/path.sh" -delegate-autorun-arguments "arg1,
arg2, arg3 with spaces, arg4"
```

Other available optional options:

- `--delegate-storage-root "path" ; e.g. "%home%/Desktop"`
- `--delegate-dcv-gl-enabled true|false; default is true`
- `--delegate-init-file "initfile"`
- `--delegate_max_concurrent_clients`
- `--delegate-enqueue true|false ; default is false`
- `--delegate-disable-retry-on-failure true|false ; default is false`

Restarting DCV Session Manager

By default, the DCV Session Manager Broker is configured with enable-persistence=false in the session-manager-broker.properties configuration file. If you restart the DCV Session Manager with this default DCV Session Manager Broker setting, the list of sessions and servers are lost until all of the DCV Session Manager agents repopulate the information for the broker.

To avoid issues with EF Portal while running, restart the DCV Session Manager Broker by taking the following steps.

1. Stop the EF Portal service and verify that the EF Portal processes are killed: `ps -f | grep enginframe`.
2. Restart the DCV Session Manager Broker.
3. Wait at least 5 minutes and make sure that all of the agents have communicated with the DCV Session Manager Broker. Verify by running the `dcvsm describe-servers` and `dcvsms describe sessions` commands.

4. Restart the EF Portal service.

You can also avoid this issue by setting enable-persistence=true in the session-manager broker.properties configuration file. For more information, see Configuring broker persistence in the *DCV Session Manager Administrator Guide*.

Automatic recovery of sessions available in DCV Session Manager

In certain situations (disk space full, user id changes, …) it can happen that EF Portal is missing information about sessions available in the DCV Session Manager. We have added automatic recovery of running sessions available in the DCV Session Manager Broker to EF Portal to mitigate such typically rare situations.

If there is no other session of the user then the user has to create a session to recover the other session as EF Portal will not be active when there is no session of a user.

The main reason for this feature is that it is not easy to recreate a DCV SM session in EF Portal so this enables a way to recover a session automatically.

Plugin limitations

Custom Init Scripts aren't supported. Linux virtual sessions use the default scripts that are provided by DCV.

Sessions can be shared only with Collaborators. There is currently no support for sharing with Viewers.

Troubleshooting

Compatibility with DCV versions

EF Portal 2020.1 added several new features. These included node monitoring, Linux console sessions, autorun files, and session screenshots. These features only work with DCV 2021.0 and later. They do not work with earlier versions. To use these features and avoid issues of incompatibility, upgrade to a compatible version.

If you try to use these features with an earlier version, an error like the following might occur.

```
SMClient.getSessionScreenshot: If you are not using DCVSM 2021.0+,
please be aware that getSessionScreenshot is not supported.
Protocol version: 2020.2
```

Log files

The main DCV Session Manager plugin log file is located under the EF Portal log directory that's named `${EF_LOGDIR}/dcvsm.log`.

Managing AWS HPC Connector

HPC Connector requires AWS CLI version 2 and AWS ParallelCluster 3.0.2 or later to be pre-installed on the node that's running the EF Portal server. This means that the user efnobody running the EF Portal Server must have access to the package and permissions to run the

pcluster command.

Before you install EF Portal, make sure that AWS ParallelCluster is installed by running the following command as the user efnobody that is running the EF Portal server.

```
$ pcluster version
```

The output is as follows:

```
{
 "version": "3.2.2"
}
```

For instructions on how to set up and configure AWS ParallelCluster, see the AWS ParallelCluster guide. This setup includes setting up required dependencies. Among these dependencies, NodeJS must be accessible to the user efnobody that's running the EF Portal server.

We recommend that you install AWS ParallelCluster in a Python virtual environment. This avoids requirement version conflicts with other Python packages.

Activating an AWS ParallelCluster virtual environment

You must reactivate the virtual environment with every restart. As a result, we recommended that you add it in your init environment file (for example, `~/.bashrc`) for the user.

```
source [YOUR_VIRT_ENV_PATH]/bin/activate
```

Before installing EF Portal with AWS HPC Connector

To start HPC Connector, the EF Portal installer requires the following parameters:

- An AWS Region where the cluster is created and managed.
- An Amazon S3 bucket to transfer data between EF Portal and the remote clusters.
- Three IAM roles that are used to access data on the S3 bucket, submit jobs using SSM, and manage clusters using AWS ParallelCluster.
- An AWS instance profile or an AWS profile name containing AWS credentials for an IAM user (<ef iam-user>) able to assume the roles.

To simplify the setup of these resources, you can use ready-made deployment scripts to create the required resources on your AWS account.

AWS credentials and profile

HPC Connector requires users to have a valid set of credentials for <ef-iam-user> that are stored in the .aws/credentials file of the user efnobody that's running the EF Portal server. Make sure that the credential file has an entry that's similar to the following example.

```
[<profile-name>]
aws_access_key_id=<aws access key for the account>
aws_secret_access_key=<aws secret access for the account>
```

EF Portal assumes that this credential file is already present at installation time, and that a profile

with the name that's provided during setup is already present. It asks for the profile name that's used for creating the clusters, and performs a check on the credentials file. If no file is present or if a profile with the given profile name is not present, the installer logs a warning and continues its operation, assuming that the profile will be created at a later time.

Amazon S3 bucket for data transfer

When launching a job on a cluster running on AWS, HPC Connector uses an Amazon S3 bucket to transfer input and output data to the remote folder where the job is to run. To use the Amazon S3 bucket, you must provide the Amazon resource name (ARN) of the S3 bucket when using the EF Portal installer.

Warning

- In general, the overall size of a file that's transferred for a single job doesn't need to exceed 100MB. If the overall file size exceeds 100MB, the system might time out on the transfer operation, and the job or copy might fail. If your jobs need to access larger datasets, make sure that these are available to the cluster through a shared file system. For more information about how to mount shared storage, see Shared storage in the AWS ParallelCluster User Guide.
- HPC Connector doesn't support Using Amazon S3 bucket keys.

HPC Connector IAM roles

AWS HPC Connector uses AWS Identity and Access Management (IAM) roles to control permissions that are associated with the AWS resources that are deployed to a selected AWS account. HPC Connector runs multiple operations on the selected AWS account. It creates and manages clusters using AWS ParallelCluster commands. It also submits and manages jobs on remote clusters, and transfers input and output data between clusters and the EF Portal spoolers. HPC Connector uses different roles to limit the scope to the minimum permissions required for each operation.

- An IAM role for managing AWS ParallelCluster (ef-parallelcluster-role) in your account. This role is used by HPC Connector for creating, starting, and stopping clusters. It must have AWS ParallelCluster's policies attached for managing the clusters.
- An IAM role for accessing the Amazon S3 bucket (ef-s3-role). This is used by HPC Connector for transferring the data back and forth from the remote destination. This role has a policy (ef-s3-role policy) that's attached for accessing the Amazon S3 bucket and reading or writing objects.
- An IAM role for running jobs remotely using SSM (ef-ssm-role). This is used for launching job scripts remotely on the clusters. This role has a policy (ef-ssm-role-policy) that's attached for sending commands to the remote instances.

You can create IAM policies and roles to control the permissions that are granted to HPC Connector and to the users of the cluster. The following examples show the IAM policies and roles that are required to use HPC Connector. In the policies, replace aws-account-id and similar strings with the appropriate values.

Note
The following examples include Amazon Resource Names (ARNs) for the resources. If you're working in the AWS GovCloud (US) or AWS China partitions, the ARNs must be changed. Specifically, they must be changed from "arn:aws" to "arn:aws-us-gov" for the AWS GovCloud (US) partition or "arn:aws-cn" for the AWS China partition. For more information, see Amazon Resource Names (ARNs) in AWS GovCloud (US) Regions in the *AWS GovCloud (US) User Guide* and ARNs for AWS services in China in *Getting Started with AWS services in China*.

AWS ParallelCluster IAM policy and role

HPC Connector uses AWS ParallelCluster to provision clusters and must have access to both the Base user policy required to invoke AWS ParallelCluster features and the policy for Privileged IAM access mode.

For more information how to create the policy, see AWS Identity and Access Management roles in AWS ParallelCluster 3.x.

The AWS ParallelCluster IAM role then needs to be configured with the following trust relationship to allow EF Portal to use it:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::<aws-account-id>:user/<ef-iam-user>"
 },
 "Action": "sts:AssumeRole"
 }
 ]
}
```

SSM IAM policy and role

The following policy shows the permissions that are required by HPC Connector to run SSM commands for managing jobs.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:SendCommand"
 ],
 "Resource": [
 "arn:aws:ec2::<aws-account-id>:instance/*"
 ],
 "Condition": {
 "StringLike": {
 "ssm:resourceTag/parallelcluster:node-type": ["HeadNode"]    }
 }
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:SendCommand"
 ],


 "Resource": [
 "arn:aws:ssm:::document/AWS-RunShellScript"
 ]
 },
```

```
 {
 "Action": [
 "ssm:GetCommandInvocation"
 ],
 "Effect": "Allow",
 "Resource": "*"
 }
 ]
}
```

Create the policy, assign it a name (ef-ssm-role-policy) and attach it to an IAM role (for example, ef-ssm-role). The policy requires all the instances to have a specific tag that's assigned automatically by AWS ParallelCluster to head nodes. This prevents arbitrary SSM commands from running on instances that aren't head nodes. Then configure it with the following trust relationship to allow EF Portal to use it:

```
 {
  "Version": "2012-10-17",
  "Statement": [
  {
  "Effect": "Allow",
  "Principal": {
  "AWS": "arn:aws:iam::<aws-account-id>:user/<ef-iam-user>"
  },
  "Action": "sts:AssumeRole"
  }
  ]
 }
```

Amazon S3 IAM policy and role

The following policy shows the permissions that are required by HPC Connector to move files between the EF Portal node and remote spoolers using Amazon S3. Replace <s3-bucket> with the actual Amazon S3 bucket name.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "s3:ListBucket"
 ],
 "Resource": [
 "arn:aws:s3:::<s3-bucket>"
 ],
 "Effect": "Allow"
 },
 {
 "Action": [
 "s3:PutObject",
 "s3:GetObject",
 "s3:DeleteObject"
 ],
 "Resource": [
```

```
 "arn:aws:s3:::<s3-bucket>/*"
 ],
 "Effect": "Allow"
 }
 ]
}
```

Create the policy, assign it a name (for example, ef-s3-role-policy), and attach it to an IAM Role (ef-s3-role). Then configure it with the following trust relationship to allow EF Portal to use it:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::<aws-account-id>:user/<ef-iam-user>"    },
 "Action": "sts:AssumeRole"
 }
 ]
}
```

Access to the Amazon S3 bucket is required only by the head nodes of the remote clusters. Therefore, you must add a condition that limits the access only to the instance policies that are created by AWS ParallelCluster. This can be done by specifying a condition on the IAM role (ef-s3- role).

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::<aws-account-id>:root"
 },
 "Action": "sts:AssumeRole",
 "Condition": {
 "StringLike": {
"AWS:PrincipalArn":
"arn:aws:iam::<aws-account-id>:role/parallelcluster/*"    }
 }
 }
 ]
}
```

EF Portal IAM user

An <ef-iam-user> must be configured with the following policy that allows it to assume the previously defined AWS ParallelCluster, Amazon S3, and SSM roles.

```
{
 "Version": "2012-10-17",
```

```
 "Statement": [
 {
 "Action": "sts:AssumeRole",
"Resource":
"arn:aws:iam::<aws-account-id>:role/<ef-parallelcluster-role>",
"Effect": "Allow"
 },
 {
 "Action": "sts:AssumeRole",
 "Resource": "arn:aws:iam::<aws-account-id>:role/<ef-s3-role>",
"Effect": "Allow"
 },
 {
 "Action": "sts:AssumeRole",
"Resource": "arn:aws:iam::<aws-account-id>:role/<ef-ssm-role>",
"Effect": "Allow"
 }
 ]
}
```

AWS ParallelCluster configuration requirements

Topics

Required additional IAM policies

AWS HPC Connector requires AWS ParallelCluster configurations to have specific additional policies attached to the IAM role used for the head node. Most specifically, as the job submission uses SSM, the AmazonSSMManagedInstanceCore must be attached to the additional policies. In addition to SSM, the head node must also assume the role that's needed by HPC Connector to transfer the files back and forth.

AWS ParallelCluster head node policy

AWS ParallelCluster configurations must include a role policy for the cluster head node (parallelcluster-ef-instance-policy) to allow data transfer to and from the regional Amazon S3 bucket. Be aware that the name of the policy must start with the parallelcluster prefix in order for it to be attached to the head node.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": "sts:AssumeRole",
 "Resource": "arn:aws:iam::<AWS ACCOUT ID>:role/<ef-s3-role>",
"Effect": "Allow"
 }
 ]
```

```
}
```

The policies (both the head node policy and the SSM managed instance core policy) must be present in your configuration in order for the cluster to be managed by HPC Connector

```
HeadNode:
 Iam:
 AdditionalIamPolicies:
 - Policy: arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
 - Policy:
arn:aws:iam::<account-id>:policy/<parallelcluster-ef-instance-policy>
```

Scoping down HPC Connector

HPC Connector allows running SSM commands only to head nodes. It does this by using the parallelcluster:node-type tag added automatically by AWS ParallelCluster when it provisions a new head node instance. This restriction is specified in the SSM policy that is required by HPC Connector to work properly. However, the default policy HPC Connector can access any head node, even if it wasn't created specifically by HPC Connector itself. This can be further scoped down by specifying a different tag name and value in the SSM policy (ef-ssm-role-policy) and having AWS ParallelCluster set that tag on the head node by adding a Tags section in the configuration file. For instance, for setting a tag named ef:instance to the remote value, the following section can be added in AWS ParallelCluster's configuration file.

```
Tags:
 -Key: ef:instance
 Value: remote
```

Managing clusters

The management of clusters in EF Portal is performed within two distinct views that are available on demand in the Admin View section of EF Portal.

- The AWS Cluster Configurations view, which manages AWS ParallelCluster configurations.
- The Clusters view, which shows all of the running clusters accessible by EF Portal.

AWS Cluster Configurations page

The **AWS Cluster Configurations** page displays all the AWS ParallelCluster configurations that administrators have imported in EF Portal. The configurations define all the properties of a cluster on AWS, including queues, compute resources, and shared storage. HPC Connector only supports AWS ParallelCluster 3 template files as cluster configurations.

In this view administrators can register new cluster configurations, rename and delete them, and launch new clusters based on that configuration. The following sections will provide instructions on how to operate on the AWS Cluster Configurations page.

Note: Starting from EF 2025.2, the **AWS Cluster Configurations** service is not visible by default, if you're interested on this feature you need to uncomment the `<ef:service id="manage.aws-cluster-configurations">` in the `$EF_ROOT/plugins/applications/WEBAPP/applications.admin.xml` file.

List the AWS Cluster Configurations

- Log in to the Workspace as a user with administrator privileges.
- To open the Admin view, on the top-right corner choose **Switch to Admin View**.
- Under the **Manage** section in the left menu, select **AWS Cluster Configurations**.

Register an AWS Cluster Configuration

Note
Before registering an AWS ParallelCluster configuration, make sure it complies with the HPC Connector requirements. For more information, see AWS ParallelCluster configuration requirements.

- Log in to the Workspace as a user with administrator privileges.
- To open the Admin's portal, on the top-right corner choose **Switch to Admin View**.
- Under the **Manage** section in the left menu, select **AWS Cluster Configurations**.
    - The list of the cluster configurations is displayed in the table.
- Choose **Register** at the top of the table.
- In the **Register Cluster Configuration** box, fill in the fields:
    - **AWS ParallelCluster configuration** - Choose between "Upload file" and "Paste text"
    - **Upload file** - Select a valid AWS ParallelCluster configuration file
    - **Paste text** - Enter a valid AWS ParallelCluster configuration file in the textbox.
- Choose **Register**.

Rename an AWS Cluster Configuration

- Log in to the Workspace as a user with administrator privileges.
- To open the Admin's portal, on the top-right corner choose **Switch to Admin View**.
- Under the **Manage** section in the left menu, select **AWS Cluster Configurations**.
    - The list of the cluster configurations is displayed in the table.
    - Select the **Actions** drop-down menu on your chosen configuration.
- Choose **Rename**.
- Add the new configuration name.
- Choose **Rename** to submit the new name. The new name will be visible in the configuration list after a few moments.

Delete an AWS Cluster Configuration

- Log in to the Workspace as a user with administrator privileges.
- To open the Admin's portal, on the top-right corner choose **Switch to Admin View**.
- Under the **Manage** section in the left menu, select **AWS Cluster Configurations**.
    - The list of the cluster configurations is displayed in the table.
- Select the **Actions** menu on your chosen configuration.
- Choose **Delete** action from the dropdown.
- Choose **OK** to confirm deletion. The configuration will be removed from the configuration list after a few moments.

Start a new cluster from an AWS Cluster Configuration

- Log in to the Workspace as a user with administrator privileges.
- To open the Admin's portal, on the top-right corner choose **Switch to Admin View**.
- Under the **Manage** section in the left menu, select **AWS Cluster Configurations**.
    - The list of the cluster configurations is displayed in the table.
- Select the **Actions** menu on your chosen configuration.
- Choose **Create Cluster** action from the menu.

- Enter the **Cluster Name** field with your preferred cluster name.
- For **User Mode**, choose **Single user** or **Multi User 1:1**.
- Choose **Create** to submit the cluster creation.
- Creating the new cluster might require a few minutes. During that time, the state of the cluster in the cluster list page will change upon refresh. When a cluster is ready to be used its status will transition to **Ready** on the **Clusters** page.

Clusters page

The Clusters view contains all the clusters from all the grid managers that are linked to EF Portal. For each cluster, we show the name, the scheduler, the type (Local or AWS Cloud), the list of available queues, and the cluster status. Only clusters with the status Ready can serve traffic and be used for job submission.

By choosing the **Actions** arrow that appears hovering on a row, some functionalities are shown on clusters with type AWS Cloud. Depending on the state of the cluster, you can choose one or more actions between Start, Stop, Delete, Rename, and Share.

- The **Start** action is used to start the head node of a specific cluster.
- The **Stop** action is used to stop the head node of a specific cluster. A stopped cluster can't serve traffic until it is started again.
- The **Delete** action is used to completely delete all the resources related to the cluster itself. After a cluster is deleted, it isn't possible to recover it. A new cluster must be created from the same configuration. Take extra caution when deleting a cluster, and ensure that data has been appropriately synced prior to deleting a cluster.
- The **Rename** action is used to redefine the label associated with the cluster. This label is used to refer to the cluster in other EF Portal views, such as the Service Editor list. Changing the cluster label doesn't affect the cluster operation or the related AWS CloudFormation stack. Changing the label automatically updates any service that had previously referred to it as well.
- The **Share** action gives visibility of a cluster to a specific set of users or groups. By default, only Administrators can see AWS Cloud clusters. When choosing **Share**, a dialog opens where you can select which users can submit jobs on the cluster. After choosing **Share**, the previous share settings are overridden. More specifically, you can choose one of the following options:
    - Only Administrators
    - All Users
    - Selected groups (here you must select one or more of the groups that you see in the specific area of the dialog.)

List the existing clusters

- Log in to the Workspace portal as a user with administrator privileges.
- In the menu, select **Switch to Admin View**.
- In the navigation pane, under **Infrastructure**, select **Clusters**.

If the cluster is an AWS ParallelCluster cluster, its type is AWS Cloud. The allowed actions might vary based on the type and current status of the cluster.

Start a cluster

- Log in to the Workspace portal as a user with administrator privileges.
- In the menu, select **Switch to Admin View**.
- In the navigation pane, under **Infrastructure**, select **Clusters**.
- Select the **Actions** menu on your chosen configuration.
- Choose **Start**.

After the **Start** action is selected, users can't see the cluster in their list until the cluster transitions into the **Ready** state.

Stop a cluster

- Log in to the Workspace portal as a user with administrator privileges.
- In the menu, select **Switch to Admin View**.
- In the navigation pane, under **Infrastructure**, select **Clusters**.
- Select the **Actions** menu on your chosen configuration.
- Choose **Stop**.

The cluster isn't available to the users immediately. After a few moments, it transitions into the **Stopped** state.

Delete a cluster

- Log in to the Workspace portal as a user with administrator privileges.
- In the menu, select **Switch to Admin View**.
- In the navigation pane, under **Infrastructure**, select **Clusters**.
- Select the **Actions** menu on your chosen configuration.
- Choose **Delete**.

Note
The cluster must be in a **Stopped** state before it's deleted.

- Choose **Yes** on the confirmation dialog.
- The selected cluster is deleted after a few minutes, and all its resources are removed. After this is complete, the cluster transitions into the **Deleted** state. By default, the entry is removed from the list after one day.

Share a cluster

- Log in to the Workspace portal as a user with administrator privileges.
- In the menu, select **Switch to Admin View**.
- In the navigation pane, under **Infrastructure**, select **Clusters**.
- Select the **Actions** menu on your chosen configuration.
- Choose **Share**.
    - A window opens where you can select users to share the cluster with.
- Specify the users to share the cluster with: Only Administrators, All Users, or Selected Groups.
- Choose **Save**.
- When the dialog is closed, the new configuration is applied to the cluster, and it's visible to the specified users or groups depending on its state.

Note
Only clusters in the **Ready** state are visible to the users.

Rename a cluster

- Log in to the Workspace portal as a user with administrator privileges.
- In the menu, select **Switch to Admin View**.
- In the navigation pane, under **Infrastructure**, select **Clusters**.
- Select the **Actions** menu on your chosen configuration.
- Choose **Rename**.
- Enter a new cluster name in the window.
- Choose **Rename**.
- The list of clusters is updated after a few moments with the new name chosen for the

cluster.

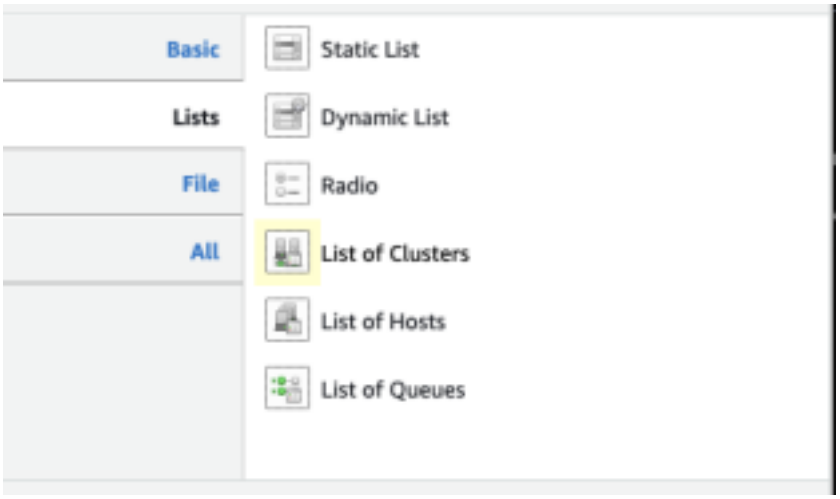
- In the **Rename Cluster** box, enter the preferred name in the **Cluster Name** field. • Choose **Rename**.

Current limitations

HPC Connector supports only jobs that are related to the same cluster in a spooler.

Managing jobs

To manage a job using HPC Connector, you must specify the cluster where you want to submit a given job. For this purpose, the service editor allows users to select a cluster by means of the cluster list component. The component shows the list of clusters that are shared with a given user.



On the action script part of your service, the usual `application.submit` action is required to have a `--cluster` option where the selected cluster can be specified. For HPC Connector to submit jobs remotely, the `--jobmanager` parameter must be set to the hpc value.

A cluster can be specified programmatically using the cluster ID rather than the name. The cluster ID information is available in the details page of a cluster. This page is reachable by selecting the name for a given cluster in the clusters page.

Validating a remote job submission

HPC Connector has a mechanism that copies the local data that's required to submit a job remotely. However, this mechanism is limited to the use of small files, which can't exceed 100MB overall.

If you run jobs with different requirements in terms of file size, a different approach is required depending on the specific scenario. AWS provides several solutions for synchronizing data on-premises remotely. Solutions such as AWS DataSync or AWS Storage GateWay, for example, can be used for creating tailor made solutions.

Regardless of the synchronization approach adopted, it might happen that the content required on the cloud isn't up to date with the equivalent content on-premises due to eventual consistency. In these scenarios, it's useful for a service to fail fast upon a validation check performed before submitting the job to a remote cluster.

The `HPCC_VALIDATION_SCRIPT_PATH` environment variable allows administrators to specify the path of an executable that is launched before a job is submitted remotely. The executable has access to all the environments set for the job and therefore can validate if a given job has all the necessary resources for running properly: project files, users, permissions.

A validation script is required to return one of two states.

- `SUCCESS` indicates that the validation succeeded and that the job is therefore allowed to continue its submission process remotely.
- `FAILURE` indicates an issue in the synchronization and cancels the job submission. A canceled job can then be repeated by the user at a later time.

Any value that's returned different from SUCCESS or FAILURE is interpreted by HPC Connector as a failure and HPC Connector cancels the job submission.

Note
The validation script is meant to perform only a validation of the requirements for job submission. It's not meant as a way for transferring data from on-premises. Due to its scope, it's meant to be run in a short time frame that's no more than 10 seconds. Having a validation script that takes near to the 10 second quota to run might cause performance issues in the EF Portal that impacts the whole system.

Customizing logging

Logging is an integral component to any software development project. During the development stages it offers a valuable source of debugging information for the developer. During deployment it can provide valuable operational data that allows administrators to diagnose problems as they arise.

Tomcat® logging

EF Portal comes with an Apache Tomcat® servlet container. Tomcat® log files are the first source of information concerning EF Portal's status. Log files are located on EF Portal Server's host under `$EF_TOP/logs/<HOSTNAME>/tomcat`.

These are the most interesting log files:

- `catalina.out`

  Contains Tomcat®'s Java™ process standard output and standard error. Tomcat® startup and shutdown messages are written here.
- `catalina.[yyyy]-[mm]-[dd].log`

  Contains Tomcat® and its libraries logging information on a day-by-day basis.
- `localhost_access_log.[yyyy]-[mm]-[dd].txt`

  Contains all web accesses on a day-by-day basis. Any resource served by Tomcat is logged here with information about the amount of transferred data. Also HTTP status codes like 404 Not Found, 403 Forbidden are logged here.
- `efportal.[yyyy]-[mm]-[dd].log`

  Contains Tomcat® error logs about EF Portal Portal not caught by EF Portal itself.

The standard Tomcat® configuration fits the most common installations. In case of specific needs you can modify the default `$EF_TOP/<VERSION>/enginframe/conf/logging.properties` configuration. Refer to the official Tomcat® documentation for more details.

EF Portal server and agent logging

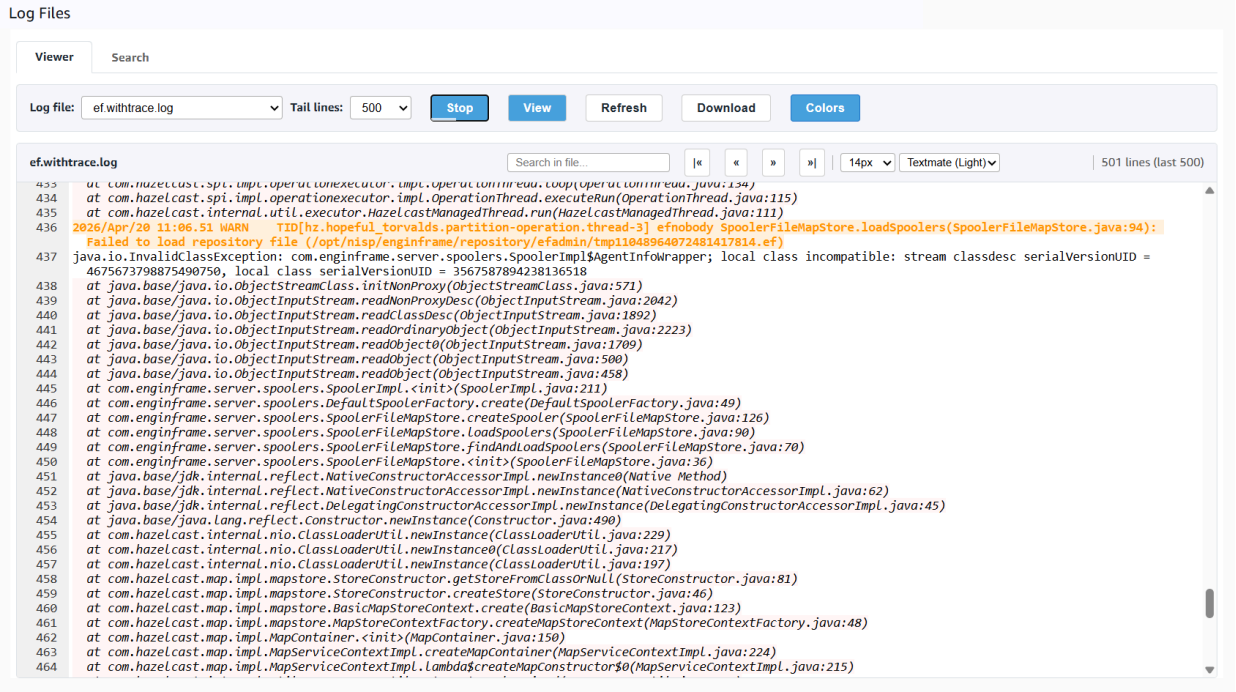
Topics

Log files viewer

Starting from EF Portal 2026.0, log files can be reviewed through the “View Log Files” service available in the Troubleshooting section of the Operational Dashboard.

The log files viewer introduces a fully themed interface with integrated navigation and advanced search capabilities. It supports cross-file searches, including predefined keywords, and provides direct links to each match to streamline troubleshooting.

The viewer also offers a tail mode with automatic refresh, color-coded log levels for easier interpretation, and multi-host log browsing. Additionally, results from cross-file keyword searches can be exported in CSV format for further analysis.



Log Configuration files

The default logging configurations are located under `$EF_TOP/<VERSION>/enginframe/conf`:

- log.server.xconf

  Contains server configuration.
- log.agent.xconf

  Contains agent configuration.

Both are XML files with the same syntax.

In case you need to modify the defaults, you can specify a new configuration in the `log.server.xconf` and `log.agent.xconf` files under the configuration directory `$EF_TOP/conf/enginframe`.

In these files you can configure the location where you store log files, their verbosity level, and rotation policies. By default, EF Portal is configured to write only warning and error messages and to keep a history of 4 log files with a 10 MB maximum size.

The log's default location is under `$EF_TOP/logs/<HOSTNAME>` on server and agent hosts:

- `ef.log`

  Contains the server's logging. It also contains the local agent's logs. It is located on the server's host.
- `agent.remote.stdout`

  Contains the agent's process standard output. It is located on the agent's host.
- `agent.remote.stderr`

  Contains the agent's process standard error. It is located on the agent's host.
- `agent.remote.log`

  Contains the agent's process logging. It is located on the agent's host.

If you need to quickly switch the log configuration to produce verbose logging, you can use `log.server.verbose.xconf` and `log.agent.verbose.xconf` files located in `$EF_TOP/<VERSION>/engifnrame/conf`.

You need to backup the previous `log.server.xconf` and `log.agent.xconf` files and replace them with the verbose configurations.

We don't recommend you use a verbose logging configuration when you're in a production environment. The performance impact of logging can be significant, depending on the logging threshold that is configured and especially if a large number of users are accessing the portal.

Apache® Log4j2 log configuration

EF Portal is configured for Apache® Log4j2 logging as described in the following paragraphs.

EF Portal server Log4j2 logging is configured in `${EF_TOP}/<VERSION>/enginframe/conf/log4j2.server.xml`. By default, logs are written to `${EF_LOGDIR}/ef.log`.

EF Portal agent Log4j2 logging is configured in `${EF_TOP}/<VERSION>/enginframe/conf/log4j2.agent.xml`. By default. logs are written to `${EF_LOGDIR}/agent.remote.log`.

Log4j2 log configuration files apply to any EF Portal internal dependency that uses Log4j2 logging, such as MyBatis or Quartz.

If you need to modify the defaults, you can specify a new configuration in the `log4j2.server.xml` and `log4j2.agent.xml` files under the configuration directory

$EF_TOP/conf/enginframe.

Change log file locations

The EF Portal logging system gives you complete flexibility on where to store log files. For example you may want to move log files from their default location to store them on a high speed file-system or to conform to your company's policies.

The location configuration is done in the <targets> section by setting the <filename> tag. The text inside this tag represents the path to the log file.

For example this XML text sets ${EF_LOGDIR}/ef.log for the core components:

```
<targets>
 <efportal id="core">
 <filename>${EF_LOGDIR}/ef.log</filename>
 ...
```

You can also decide whether to append or overwrite existing files by using the <append> tag. In the following example the log file will be overwritten on every server restart:

```
<targets>
 <efportal id="core">
 <filename>${EF_LOGDIR}/ef.log</filename>
 <append>false</append>
 ...
```

Change log file size and the rotation policy

Each EF Portal Server and EF Portal Agent is configured to write log files up to 10 MB and then to create another file to a maximum of 4 files. When a new log file is written the oldest one is deleted. This configuration guarantees that each server and agent uses no more than 40 MB of disk space for log files. You may want to change this rotation policy for example because you need more history. A change may also be helpful if you have increased log verbosity.

Changing the <rotation> tag inside the <targets> section configures the rotation policy. The max attribute defines the maximum number of files, while the <size> tag contains each file's size.

For example the following XML text defines a rotation of 5 files each one of 5 MB:

```
<rotation type="revolving" max="5">
 <size>5m</size>
</rotation>
```

Change log level

EF Portal logging allows one to have a fine grain control over which statements are printed. In a development environment, you might want to enable all logging statements while in a production environment it is suggested to disable debug messages to avoid performance issues. You can configure this behavior by setting the appropriate *log level*. A *log level* describes the urgency of a message. Below is a list of log levels that are usable within the EF Portal logging system:

- NONE: No messages are emitted.
- DEBUG: Developer-oriented messages, usually used during development of the product.

- INFO: Useful information messages such as state changes, client connection, and user login.
- WARN: A problem or conflict has occurred but it may be recoverable, then again it could be the start of the system failing.
- ERROR: A problem has occurred but it is not fatal. The system still functions.
- FATAL_ERROR: Something caused the whole system to fail. This indicates that an administrator should restart the system and try to fix the problem that caused the failure.

Each logger instance is associated with a log level. This allows you to limit each logger so that it only displays messages greater than a certain level. So if a DEBUG message occurred and the logger's log level was WARN, the message would be suppressed.

Changing the log-level attribute of a <category> tag sets log verbosity for the associated category. For example, this XML tag defines a default category whose log-level is INFO:

```
<category name="" log-level="INFO">
 <log-target id-ref="core"/>
</category>
```

Fine tune logging

Define new categories and targets

In a complex system, it's often not enough to suppress logging based on log-level. For instance, you might want to log the network subsystem with DEBUG log-level while the simulator subsystem with WARN log-level. To accomplish this EF Portal uses categories. Each category is a name, made up of name components separated by a ".". So a category named "network.interceptor.connected" is made up of three name components "network", "interceptor" and "connected", ordered from left to right.

Every logger is associated with a category at creation. The left-most name component is the most generic category while the right-most name component is the most specific. So "network.interceptor.connected" is a child category of "network.interceptor", which is in turn a child category of "network". There is also a root category "" that is hidden.

The main reason for structuring logging namespace in a hierarchical manner is to allow inheritance. A logger will inherit its parent log-level if it has not been explicitly set. This allows you to set the "network" logger to have INFO log-level and unless the "network.interceptor" has had its log-level set it will inherit the INFO log-level.

Categories send messages to log targets. Decoupling log message generation from handling allows developers to change destinations of log messages dynamically or via configuration files. Possible destinations include writing to a database, a file, an IRC channel, a syslog server, an instant messaging client, etc.

Adding a <category> tag inside the <categories> section defines a new category. You must specify its name and log-target to which log messages are written. The name of the category acts like a filter: all messages matching the category name are sent to the specified log-target.

For example, if you want more information from EF Portal's download component you can use the category named com.efportal.server.download:

```
<category name="com.efportal.server.download" log-level="DEBUG">
 <log-target id-ref="core"/>
</category>
```

This configuration sends all download messages to the core target.
Another useful category is the deadspooler. The deadspooler category is used by EF Portal to write messages concerning spoolers that could not be deleted and were renamed as `DEAD`. NI SP Support team recommends turning this feature on. There is no overhead and it could be very useful to know why an EF Portal spooler could not be deleted. This category is activated by setting its log-level to `INFO`:

```
<category name="deadspooler" log-level="INFO">
 <log-target id-ref="deadspooler"/>
</category>
```

Change message format

Log targets that write to a serial or unstructured store (i.e., file-system or network based targets) need some method to serialize the log message before writing to the store. The most common way to serialize the log message is to use a formatter.

The format specified consists of a string containing raw text combined with pattern elements. Each pattern element has the generalized form:

```
%[+|-]#.#{field:subformat}
```

- `+|-` indicates whether the pattern element should be left or right justified (defaults to left justified if unspecified.)
- `#.#` indicates the minimum and maximum size of output, if unspecified the output is neither padded nor truncated.
- `field` indicates the field to be written and must be one of category, context, user, message, time, rtime (time relative to start of application), throwable or priority. This parameter is mandatory.
- `subformat` specifies which piece of field is interesting for log messages.

Use the <format> tag to set log message printing.

For example the following code shows format setting for the core target:

```
<efportal id="core">
 <filename>${EF_LOGDIR}/ef.log</filename>
 <format type="efportal">
   %7.7{priority}  %5.5{rtime}  [%8.8{category}]:%{message}\n%{throwable}
</format>
 ...
```

A number of examples for format and actual output follows.

- Example

  Format:

```
                                        %7.7{priority}          %5.5{rtime}
[%8.8{category}]:%{message}\n%{throwable}
```

  Output:

```
     DEBUG 123 [network.]: This is a debug message
```

- Example

  Format:

```
%7.7{priority} %5.5{rtime} [%{category}]:%{message}\n
```

  Output:

```
DEBUG 123 [network.interceptor.connected]: This is a debug message DEBUG 123 [network]: This is another debug message
```

- Example

  Format:

```
%7.7{priority} %5.5{rtime} [%10.{category}]:%{message}\n
```

  Output:

```
DEBUG 123 [network.interceptor.connected]: This is a debug message DEBUG 123 [network ]: This is another debug message
```

Emit logs with and without stack traces in separate files

This feature introduced in EF Portal 2024.1 enables 2 different logfiles to be created, using the same rules for size and rotation. The main one, e.g. ef.log, does not contain stack traces for errors anymore (or in any case a throwable is emitted by the logging code), the secondary one, e.g. ef.withtraces.log, is exactly the same but with stack traces enabled.

The bottom line is that, compared to the past, main log files are now without stacktraces.

Following is the list of log files in scope:

- `ef.log`
- `ef.withtrace.log`
- `agent.remote.log`
- `agent.remote.withtrace.log`
- `dcvsm.log`
- `dcvsm.withtrace.log`

Auditing of login and logout events

The EF Portal Audit System is able to record noteworthy events in the server workflow, using a single entrypoint which manages any audit output destination transparently.

It is designed to support multiple output sources, each managed by the corresponding backend. Each supported backend is enabled by a configuration parameter in server.conf. All enabled backends are then invoked automatically to emit the audit event triggered by the audit system entry point.

We currently provide the Logfile backend, which writes audit events into the `EF_TOP/logs/<hostname>/audit.log` file. It is controlled by the configuration parameter

ef.audit.logfile.enabled in `${EF_CONF_ROOT}/enginframe/server.conf` which is set to false by default.

Following events are recorded:

- successful logins
- failed logins
- explicit logouts
- logouts due to session expiration

All authorities are supported, including cookie based auth.

Examples of auditing lines from the logfile backend:

```
2024/Dec/11 17:36.08 [LOGIN] Login performed, user: efadmin, authority: pam, serviceUri: //com.enginframe.interactive/list.sessions
2024/Dec/11 17:36.12 [LOGOUT] Logout performed, user: efadmin, authority: pam
2024/Dec/11 17:36.14 [LOGIN_FAILED] Login failed, error: Authentication Error Missing credentials, user: efadmin, authority: pam, serviceUri: //com.enginframe.interactive/list.session
2024/Dec/11 17:36.21 [LOGIN] Login performed, user: efadmin, authority: pam, serviceUri: //com.enginframe.interactive/list.sessions
2024/Dec/11 17:46.23 [LOGOUT_EXPIRED] Logout performed due to session expiration, user: efadmin, authority: pam
```

EF Portal scriptlet logging

EF Portal modularity lets developers add new features by deploying their plugins. If your plugin uses scriptlets, you have the same logging support on which EF Portal is based. As an administrator you must know how to set up scriptlet logging for both development and production environments.

Creating `$EF_TOP/<VERSION>/enginframe/plugins/<myplugin>/conf/log.xconf` or `$EF_TOP/conf/plugins/<myplugin>/log.xconf` on EF Portal Server's host enables scriptlet logging. This file has the same syntax as the ones shipped by EF Portal under `$EF_TOP/<VERSION>/enginframe/conf`, so all the information in EF Portal server and agent logging applies here too.

The scriptlet code can emit logging messages with any category and log level.

This is an example configuration file:

```
<?xml version="1.0"?>

<logkit>
 <factories>
   <factory
            type="efportal"
          class="com.efportal.common.utils.log.efportalTargetFactory"/>
</factories>

 <targets>
 <efportal id="plugin">
 <filename>
 ${EF_LOGDIR}/myplugin.log
```

```
 </filename>
 <format type="efportal">
 %7.7{priority} %5.5{rtime} [%{category}]:%{message}\n
 </format>
 </efportal>
 </targets>

 <categories>
 <category name="myplugin" log-level="DEBUG">

 <log-target id-ref="plugin"/>
 </category>
 <category name="" log-level="WARN">
 <log-target id-ref="plugin"/>
 </category>
 </categories>
</logkit>
```

`${EF_LOGDIR}/<myplugin>.log` contains the log messages for this configuration. It exposes only two categories: `myplugin` and the `root` one (it is the one that has an empty name.) If your code uses `myplugin` category, then it logs `DEBUG` messages; if your code uses a category that is not defined, then it logs `WARN` messages.

If the plugin `log.xconf` configuration files don't exist, then EF Portal defaults to the core `log.server.xconf` to define myplugin's logging categories.

EF Portal licenses

This chapter describes how EF Portal manages licenses, where license files are located, the meaning of license fields, how license tokens are counted, and how to check EF Portal's license token usage. This information is relevant only when running EF Portal on an on-premises or non-EC2 cloud-based server. EF Portal is licensed for free on EC2. For more information about EF Portal licensing, see Obtaining NI SP EF Portal.

License file management

EF Portal loads its license files from `$EF_TOP/license`. All files ending with .ef extension are loaded.

If you want to replace an EF Portal license while still retaining your previous license, rename your previous license by changing the `.ef` extension to something else (for example, add `.OLD` extension to the original file). If you don't make this change, EF Portal loads the old license. Having two licenses for the same EF Portal component leads to a conflict issue.

The EF Portal license management system reads licenses dynamically from the file-system, and can detect any changes you apply to licenses even to recognize if the license files were added or removed.

Configuring license files location

`$EF_TOP/license` is the default directory where EF Portal loads licenses from. This directory can be changed making EF Portal load licenses from another location in your file-system.

You can modify the `EF_LICENSE_PATH` parameter inside `$EF_TOP/conf/enginframe/server.conf`, and then define an absolute file-system path. This changes the directory EF Portal uses to load licenses.

Example: `EF_LICENSE_PATH=/mnt/server/ef-licenses`

License file format

An EF Portal license is an XML file that describes one or more EF Portal licensed components or plugins. The main component of EF Portal is EF Base. It activates the core functions of EF Portal.

The license fields are the following:

- product: This field describes the item that's being licensed (for example, EF Portal PRO)
- release: A major release of EF Portal (for example, 2026.0)
- format: The license file format number (for example, 2.0)
- component: The component that's being licensed. Example components include EF Base and rest.
- expiration: The license expiration date. The value never indicates a perpetual license.
- ip: The licensed IP addresses where EF Portal can be deployed. It can be a single host IP address, a range of IP addresses, or a list of IP addresses. The license is valid if EF Portal is running on one of the mentioned hosts IP addresses.
- type: The type of license. Valid values are DEMO, YEAR, and FULL.
- units: The number of license tokens.
- units-per-user: The number of tokens that EF Portal locks for each concurrent user.
- license-hosts: If this value is set to true, EF Portal locks a token for each host in the underlying grid infrastructure.
- hosts-preemption: If this value is set to true, EF Portal releases tokens that are received by hosts for new users that are accessing the system. This happens when all tokens have been used.
- signature: The license signature that accounts for all the license fields values.

The following is an example of an EF Portal license.

```
<?xml version="1.0"?>
<ef-licenses>
     <ef-license-group  product="EF  Portal  PRO"  release="2024.0"
format="2.0">
    <ef-license component="EF Base"
                        vendor="NI SP Software"
                        expiration="2027-08-15"
                        ip="80.20.156.116"
                        licensee="RnD Team"
                        type="DEMO"
                        units="20"
                        units-per-user="1"
                        license-hosts="true"
                        hosts-preemption="true"
                        signature="..." <!-- Omitted for space -->
    />
  </ef-license-group>
</ef-licenses>
```

License inspection

The most important license fields are component, ip, expiration, and units. You can statically verify

the first three fields. These field values depend on your EF Portal deployment setup:

- In the following example of a valid license file format, `component="EF Base"` unlocks the EF Portal core. This component is required to unlock other components, such as webservices, that can be enabled with a dedicated section in the licence file.

```
<ef-licenses>
   <ef-license-group product="EF Portal HPC PRO" release="2024"
format="2.0">   <ef-license
 component="EF Base"
 vendor="NI SP Software"
 expiration="2023-03-31"
 ip="*"


 licensee="nisp-efportal-dev"
 type="DEMO"
 units="30"
 units-per-user="1"
 license-hosts="false"
 hosts-preemption="false"
 signature="------"
 />
 <ef-license
 component="webservices"
 vendor="NI SP Software"
 expiration="2023-03-31"
 ip="*"
 licensee="nisp-efportal-dev"
 type="DEMO"
 units="30"
 signature="------"
 />
...
 </-license-group>
</ef-licenses>
```

Send questions to <support@ni-sp-software.com> for your requirements and to get help on understanding which components you actually need to satisfy your goals.

- The *IP address* depends on EF Portal's host.

EF Portal Server's Host IP Address

You must determine EF Portal Server's host IP address for a valid license.

To verify which IP address is correct for the license request, run the `ping `hostname`` command. This returns the hostname and the IP address information.

EF Portal doesn't allow for loopback IP addresses, such as 127.0.0.1. If you run a **ping** command and it returns a loopback address, you need to configure the host name resolution for EF Portal Server. You can configure it either by editing `/etc/hosts` or by changing the DNS, NIS, or LDAP configurations.

- *Expiration date* states how long your license is valid. Evaluation licenses last one month. However, NI SP can release licenses with a validity period that meets your evaluation needs. You can send a request form on the website where you download EF Portal or send an email to <support@ni-sp-software.com>.
- The *units* field expresses the total number of license tokens.

License token count

Different scenarios have to be considered when EF Portal counts license tokens. The scenarios change according to different combinations of some license fields.

License tokens are dynamically used by EF Portal on the base of system usage and load, such as the number of concurrent users or the number of hosts in the grid environment accessed through EF Portal. When EF Portal consumes a token, it subtracts it from the total number of available tokens expressed by the units license field.

Tokens that are acquired by a user are always released after the user logs out of the system or on his working session expires. When a user logs out or when a session expires, the user's tokens are released. Tokens acquired by hosts are released only on pre-emption (if expressed in the license, for the needed amount) or when EF Portal is restarted (all tokens are released).

The following scenarios depict how EF Portal acquires and releases license tokens:

1.

```
license-hosts="true"
hosts-preemption="true"
```

EF Portal acquires one token for each concurrent user accessing the system and one token for each host in the underlying computing environment.

Because pre-emption on hosts tokens is switched on, when all the tokens are used, additional users logging into the system are granted access by reclaiming one token from those acquired by hosts. The host whose token has been reclaimed is now unlicensed.

If all tokens are used and there are no more tokens to preempt, system access is denied. Available

tokens are dynamically acquired by users or hosts.

EF Portal doesn't show unlicensed hosts details, such as the kind of host, status, and memory consumption.

2.

```
license-hosts="true"
hosts-preemption="false"
```

As in the previous scenario, EF Portal acquires one token for each concurrent user accessing the system and one token for each host in the underlying computing environment.

Differently from the previous scenario, tokens that are used by hosts are never released. Only restarting EF Portal resets the host's license token status.

When all the tokens are used, the system denies further access.

Available tokens are dynamically acquired by users or hosts.

Unlicensed hosts are managed as in the previous scenario.

3.

```
license-hosts="false"
units-per-user="<n>"
```

In this scenario, EF Portal acquires n tokens for each concurrent user where n is a positive integer. No tokens are acquired by hosts meaning they are all licensed.

This kind of license fits all those cases where you deal with big or growing clusters and it is more convenient to have a flat license for the number of hosts.

When all the tokens are used by users, the system denies further access. If this happens, no token reclaim occurs because the host can't acquire new tokens.

List of licensed hosts

An optional list of licensed hosts for the previous scenarios can be created. This list spares license tokens through limiting cluster view through EF Portal.

Create license.hostlist in `EF_LICENSE_PATH` to declare the list of licensed hosts. The file must contain the host names, listed one per line. The host names must be reported in the same way that they are reported by the job scheduler.

This following is a `$EF_TOP/license/license.hostlist` example.

```
host-linux1.nisp
host-linux2.nisp
host-linux3.nisp
host-unix1.nisp
host-win1.nisp
host-win2.nisp
```

Monitoring license usage

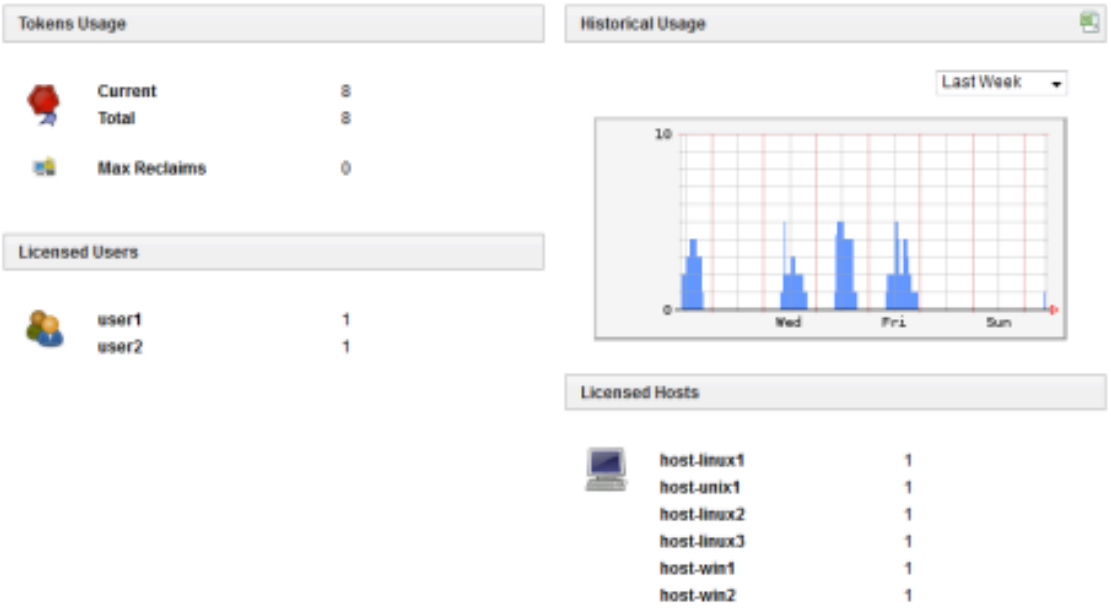
License token consumption is an important aspect for an EF Portal administrator.

The EF Portal Operational Dashboard provides administrators with an overview of installed licenses and provides details for used license tokens. Loaded licenses with their field values are shown as follows.

EF Portal License Details

The license token usage details are also provided.

EF Portal License Tokens Status

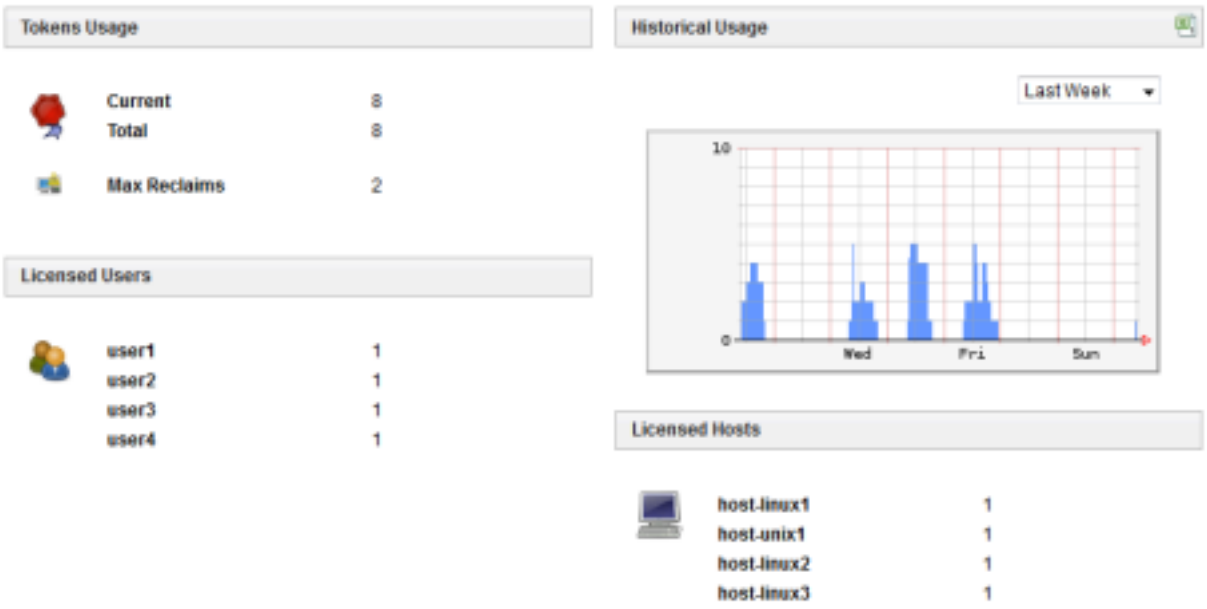


You can monitor the total number of used tokens, which users have logged in, and which hosts are currently licensed. In the preceding example, EF Portal acquired a license token for all reported users and hosts.

Note
If the same user name connects from two different workstations or browsers at the same time, an extra token is used as a result. Tokens aren't acquired nominally but on a concurrent user basis only.

You can also check if there were any token reclaims in the system previously. The value reported is the maximum number of reclaimed tokens that occurred since EF Portal's last restart.

EF Portal License Tokens Status With Reclaims



In the preceding screenshot, there are two token reclaims. In all, there are eight tokens. All eight tokens were used by six hosts and two users. The next two requests from paolo and aware were satisfied but required two token reclaims from the hosts.

Enable debug log messages for licenses

If you're experiencing problems with the EF Portal license system, require support from the NI SP Support Team, or want to check license management details, it is important to enable the license management module's debug log level.

You can enable the license module's debug log level by adding a category to the log.server.xconf file as shown in the following snippet.

```
<category name="com.efportal.common.license" log-level="DEBUG">
 <log-target id-ref="core"/>
</category>
```

Alternatively, use log.server.verbose.xconf. It already has the log level set to DEBUG. For more information, see Customizing logging.

Troubleshooting

Common issues

This section lists some common issues and the steps to resolve them:

Topics

- Breaking change in AJP Connector configuration from EnginFrame 2019.0-1424
- Interactive service imports using Slurm before and after EnginFrame version 2020
- Updating to EF Portal version 2024.0 or later

- Updating Log4j library in EnginFrame

Breaking change in AJP Connector configuration from EnginFrame 2019.0-1424

When updating to 2019.0-1424 or later from a previous version, if AJP Connector is enabled, EF Portal might fail to start after a successful upgrade. This is because of a breaking change in Tomcat starting from version 7.0.100. From Tomcat CHANGELOG:

Rename the requiredSecret attribute of the AJP/1.3 Connector to secret and add a new attribute secretRequired that defaults to true. When secretRequired is true the AJP/1.3 Connector will not start unless the secret attribute is configured to a non-null, non-zero length String.

By default, Tomcat expects secrets to be configured. If you set a requiredSecret, it's no longer picked up because of the attribute name change. To confirm an EF Portal installation is impacted by this issue, check the ${EF_LOGDIR}/tomcat/catalina.*.log files for the presence of the following message:

The AJP Connector is configured with secretRequired="true" but the secret attribute is either null or "". This combination is not valid.

Resolve the issue in the `${EF_CONF_ROOT}/tomcat/conf/server.xml` file, in the AJP Connector block. Make one of the following changes and then restart:

- If you have a non empty *requiredSecret*, rename it to *secret*
- Otherwise, add the parameter *secretRequired="false"*

Interactive service imports using Slurm before and after EnginFrame version 2020.1-r413

After the release of EnginFrame version 2020.1-r413, you can no longer import interactive services using Slurm as a grid manager with EnginFrame versions that are earlier than 2020.1-r413. To resolve this issue, you must create a new interactive service instead of importing one using Slurm.

In addition, after 2020.1-r413, the default interactive services "Linux Desktop" and "Windows desktop" don't work as-is when using Slurm as a scheduler. To make them work, open them in edit mode at least one time.

Updating to EF Portal version 2024.0 or later

Because Hazelcast is updated by 2 major versions with EF Portal version 2024.0, you must take additional steps before you update EF Portal to this and later versions.

If you have an EF Portal Enterprise setup, you must first stop the EF Portal service in all of the instances to perform a successful update to EF Portal versions 2024.0 or later.

Moreover, if you have a custom `hazelcast.xml` configuration (modified from the default configuration), you must do the following before updating to EF Portal versions 2024.0 or later:

- Port the custom `hazelcast.xml` to the format used by Hazelcast 5 following the relevant Hazelcast documentation.
- Copy it over to the `$EF_ROOT/conf` folder, overwriting the default `hazelcast.xml` file.

Updating to EF Portal version 2024.0 or later

When performing an update to this version from a previous installation of EnginFrame or EF Portal, you must make sure that the server.xml file in the tomcat folder `/opt/nisp/enginframe/conf/tomcat/conf` has this line

removed since it breaks Tomcat 9:

```
<Listener className="org.apache.catalina.core.JasperListener"/>
```

If Tomcat is configured to use HTTPS, make sure you remove the protocol parameter from the Connector section in the server.xml config file:

```
<Connector                                              port="{$httpsd_port}"
protocol="org.apache.coyote.http11.Http11Protocol"     maxThreads="150"
SSLEnabled="true" scheme="https" secure="true"
clientAuth="true" sslProtocol="TLS"
```

should be:

```
<Connector port="{$httpsd_port}"
maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
clientAuth="true" sslProtocol="TLS"
```

Updating Log4j library in EnginFrame

Due to an issue in the Apache Log4j library (CVE: https://www.randori.com/blog/cve-2021-44228) included in EnginFrame from version 2019.0-r1424 to 2021.0-r1307, it is recommended that you upgrade to the latest EF Portal version or update the Log4j library in your EnginFrame installation as described in the following instructions:

1. Identify the `EF_TOP` and `EF_ROOT` folder of your EnginFrame installation. For more information, see Installation directories.
2. Identify the log4j files:

```
$ find "$EF_TOP" -name log4j*.jar
<EF_ROOT>/agent/log4j
core-<OLD_VERSION>.jar
<EF_ROOT>/agent/log4j
api-<OLD_VERSION>.jar
<EF_ROOT>/agent/log4j-1.2-
api-<OLD_VERSION>.jar
<EF_ROOT>/WEBAPP/WEB-INF/lib/log4j
core-<OLD_VERSION>.jar
<EF_ROOT>/WEBAPP/WEB-INF/lib/log4j
api-<OLD_VERSION>.jar
<EF_ROOT>/WEBAPP/WEB-INF/lib/log4j-1.2-api-<OLD_VERSION>.jar
```

Items highlighted in *red* represent your values.

3. Stop EF Portal. For more information, see Running NI SP EF Portal:

```
$ EF_TOP/bin/enginframe stop
```

4. Backup the old log4j files to a backup directory of your choice:

```
$ EF_ROOT=...
BACKUP_AGENT_DIR=/tmp/backup/agent
BACKUP_SERVER_DIR=/tmp/backup/server
mkdir -p $BACKUP_AGENT_DIR
mkdir -p $BACKUP_SERVER_DIR
mv $EF_ROOT/agent/log4j-core-<OLD_VERSION>.jar $BACKUP_AGENT_DIR
mv $EF_ROOT/agent/log4j-api-<OLD_VERSION>.jar $BACKUP_AGENT_DIR
mv  $EF_ROOT/agent/log4j-1.2-api-<OLD_VERSION>.jar  $BACKUP_AGENT_DIR  mv
$EF_ROOT/WEBAPP/WEB-INF/lib/log4j-core-<OLD_VERSION>.jar
$BACKUP_SERVER_DIR                                                      mv
$EF_ROOT/WEBAPP/WEB-INF/lib/log4j-api-<OLD_VERSION>.jar
$BACKUP_SERVER_DIR                                                      mv
$EF_ROOT/WEBAPP/WEB-INF/lib/log4j-1.2-api-<OLD_VERSION>.jar
$BACKUP_SERVER_DIR
```

5. Download and expand the Log4j tar.gz you want to update to from https://logging.apache.org/ log4j/2.x/download.html. It's essential that you verify the integrity of the downloaded files using the PGP and SHA512 signatures, as suggested by the download page of Apache Log4j.
6. Copy the following jar files from the downloaded package to the EF Portal folder:

```
$ DOWNLOADED=apache-log4j-<NEW_VERSION>.0-bin
cp $DOWNLOADED/log4j-core-<NEW_VERSION>.jar $EF_ROOT/agent/
cp $DOWNLOADED/log4j-api-<NEW_VERSION>.jar $EF_ROOT/agent/
cp $DOWNLOADED/log4j-1.2-api-<NEW_VERSION>.jar $EF_ROOT/agent/
cp                                  $DOWNLOADED/log4j-core-<NEW_VERSION>.jar
$EF_ROOT/WEBAPP/WEB-INF/lib/  cp $DOWNLOADED/log4j-api-<NEW_VERSION>.jar
$EF_ROOT/WEBAPP/WEB-INF/lib/                                            cp
$DOWNLOADED/log4j-1.2-api-<NEW_VERSION>.jar
$EF_ROOT/WEBAPP/WEB-INF/lib/
```

7. Start EF Portal. For more information, see Running NI SP EF Portal:

```
$ EF_TOP/bin/enginframe start
```

Pushing metrics to external monitoring tools

EF Portal can push its metrics to external monitoring tools through StatsD. StatsD is a network daemon that runs on the Node.js platform and listens for statistics that are provided by tools such as counters and timers. StatsD sends aggregates to one or more pluggable backend services, such as Amazon™ CloudWatch.

We track the following metrics which can be stored into a StatsD server typically owned by the customer for further postprocessing in e.g. InfluxDB, Graphite and Grafana:

- `stats.gauges.cpu.loadAvg`
- `stats.gauges.jobs.done`
- `stats.gauges.jobs.pending`
- `stats.gauges.jobs.running`
- `stats.gauges.jobs.total`
- `stats.gauges.licenses.EF_Base.tokens`
- `stats.gauges.licenses.applications.tokens`

- stats.gauges.licenses.hpc-support.tokens
- stats.gauges.licenses.interactive.tokens
- stats.gauges.licenses.webservices.tokens
- Stats.gauges.licenses.rest.tokens
- stats.gauges.filesystem.Sessi.fsAvailInodes
- stats.gauges.filesystem.Sessi.fsAvailSize
- stats.gauges.filesystem.Sessi.fsUsedInodes
- stats.gauges.filesystem.Sessi.fsUsedSize
- stats.gauges.filesystem.Sessi.usedSize
- stats.gauges.memory.free
- stats.gauges.memory.used
- stats.gauges.spoolers.session
- stats.gauges.spoolers.total
- stats.gauges.statsd.timestamp_lag
- Stats.gauges.users.logged

Supported monitoring tools

We support every monitoring tool that implements the StatsD backend service to forward metrics. For more information about available StatsD backend services, see Supported Backends in the *StatsD README* on the GitHub website.

Prerequisites

The following are required to configure EF Portal to use StatsD to push metrics to an external monitoring tool.

- A monitoring tool that StatsD supports.
- A running StatsD instance that's configured to forward metrics to the monitoring tools in use. StatsD must be configured to use User Datagram Protocol (UDP). This is the default option. For more information about configuring StatsD, see Installation and Configuration in the *StatsD README* on the GitHub website.

Configuration

To configure EF Portal to push metrics to a StatsD instance, you must configure the `STATSD_ADDR` and `STATSD_PORT` variables. These are located in the following configuration file: `$EF_ROOT/plugins/admin/conf/admin.statistics.efconf`.

In the following example configuration, StatsD is run locally on port 8125.

```
##########################################################################
#### # StatsD Server Configuration
##########################################################################
####

STATSD_ADDR=localhost
STATSD_PORT=8125
```

Troubleshooting

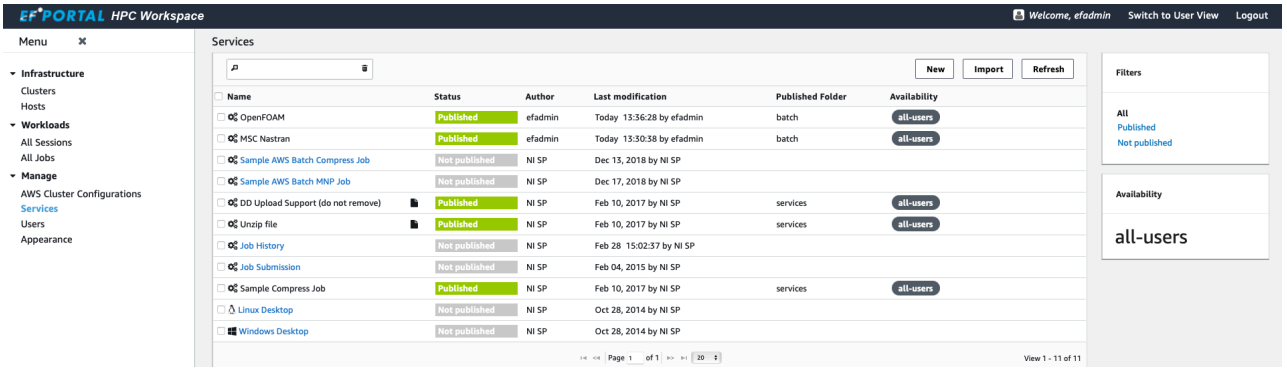
EF Portal provides the `$EF_LOGDIR/admin.log` log file. You can use it to troubleshoot and diagnose errors.

Creating and Managing Services with the Service Editor

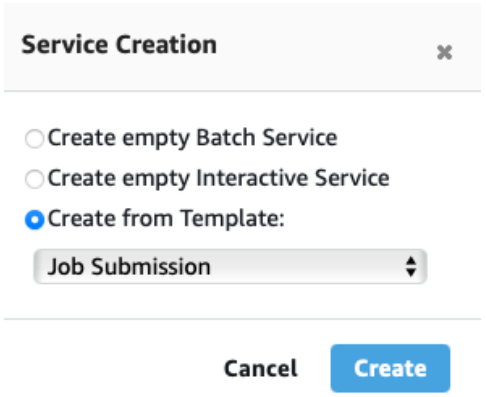
The Service Editor is an intuitive and user-friendly tool that allows you to customize and create services tailored to the specific needs of the user. You can set up these services easily with the WhatYouSeeIsWhatYouGet editor. Once created, these services are permanently saved and accessible in the Batch menu, ensuring you can quickly and efficiently manage your tasks.

How does it work?

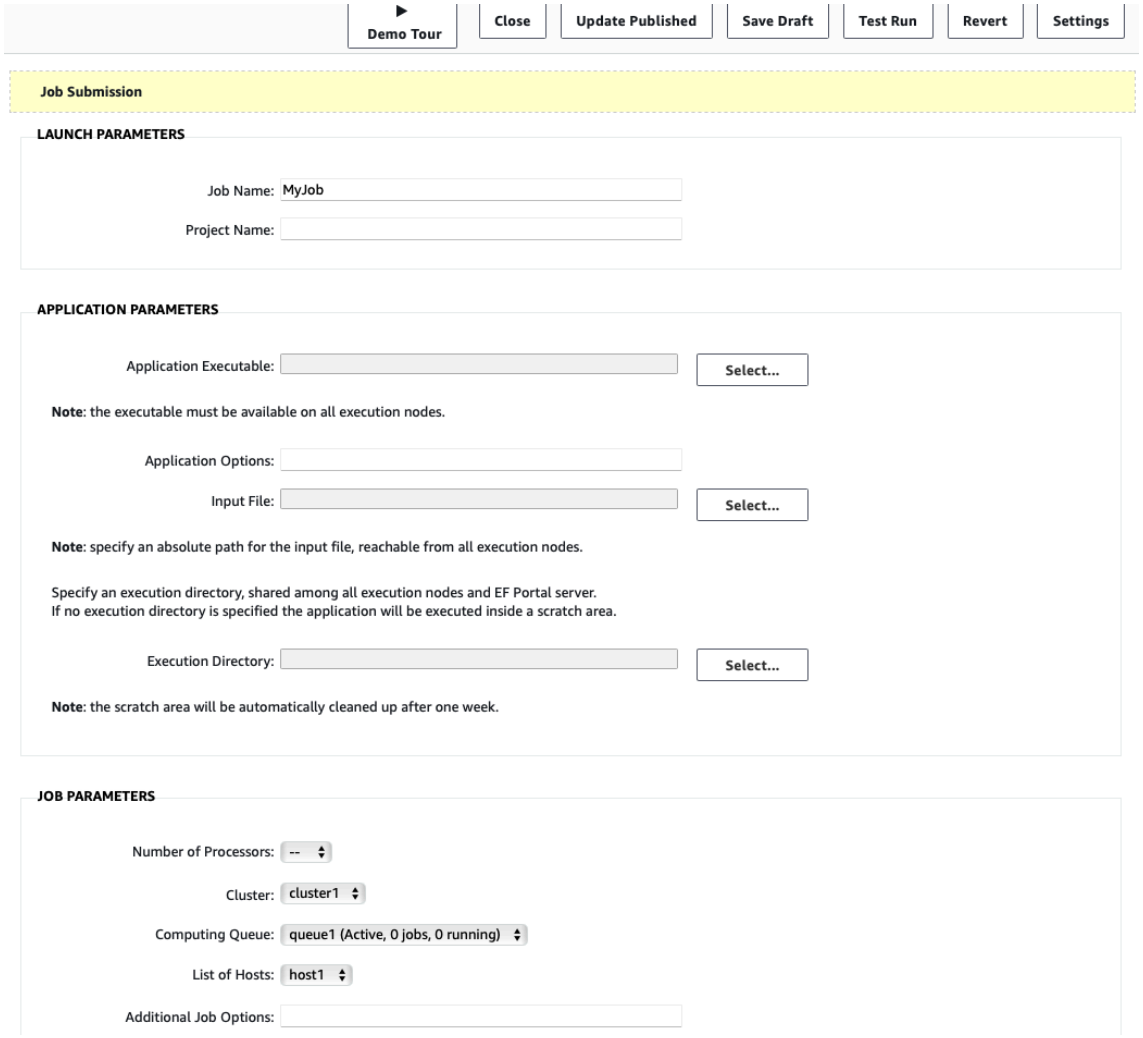
Under Menu/Manage/Services, new services can be created from the existing templates and published for users:



To create a new service from the existing templates, click on ‘new’ and then on ‘create from template’ in the drop-down menu “Service Creation” and select the desired template. In our example we have chosen “Job Submission”.

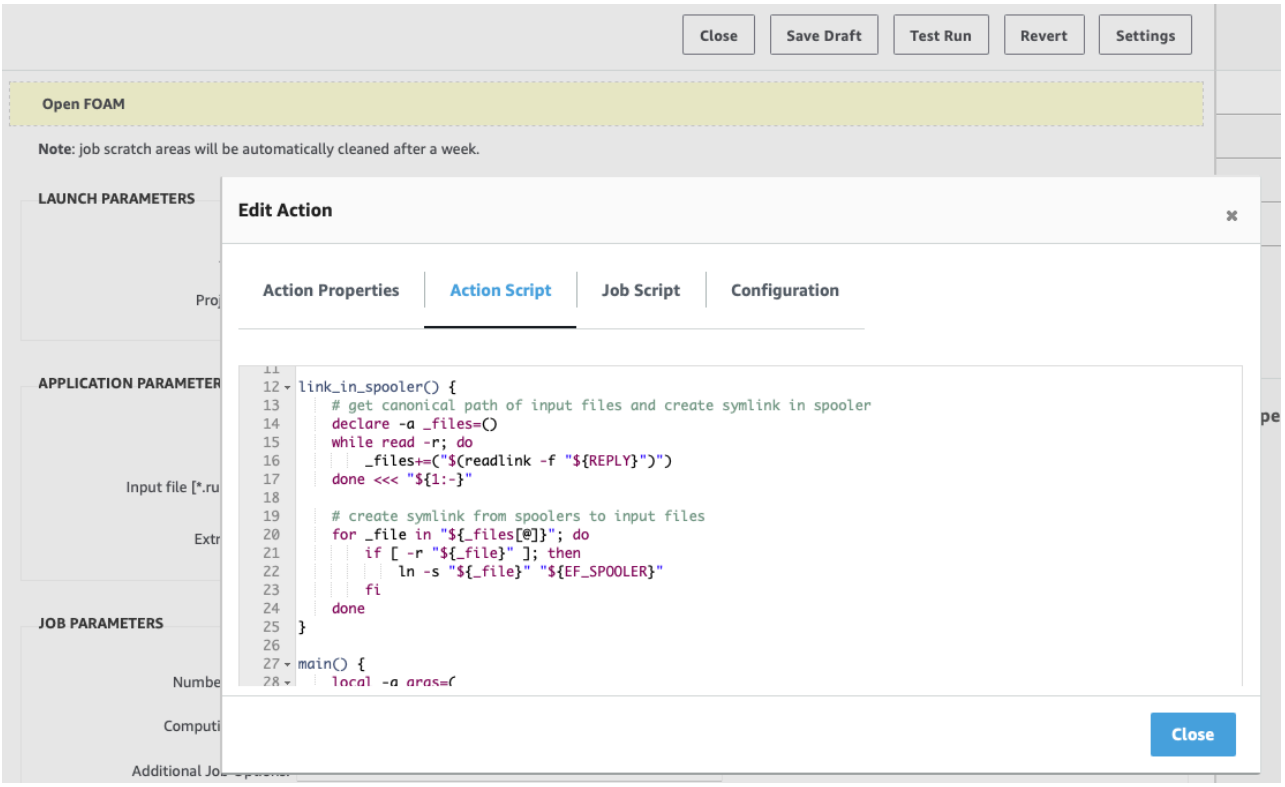


Press ”create” and this opens the user interface

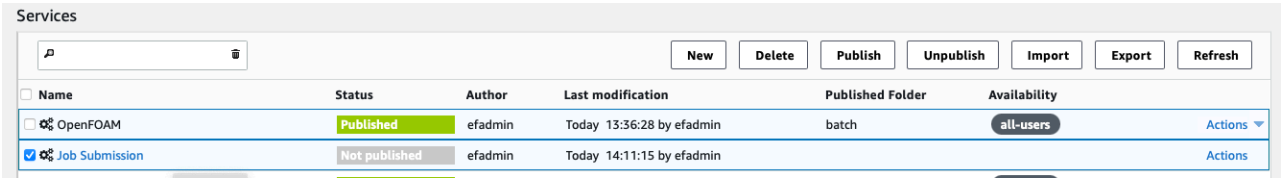


Press “Save Draft” and “Close”. This brings you directly to the services overview with our new service “Job Submission”.

Please Note: If you have specific requirements for your interactive session just open the session, press “Submit” and check out the action script. There you find a lot of different sample options which you can go through in detail and apply when you need them.



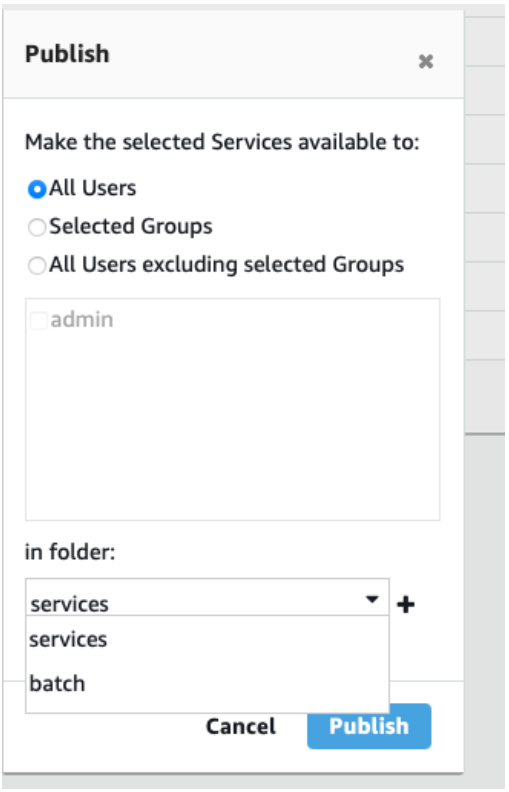
After you have created this new service, it will remain in the status “Not published” until you select the service in the checkbox and press the “Publish” button.



The services overview provides you with other features like:

- Delete: this deletes all services marked in the checkbox
- Export: this feature exports the marked service via zip file to your local machine
- Import: this feature imports a service via zip file from your local machine

When you publish the service, you have the following options:

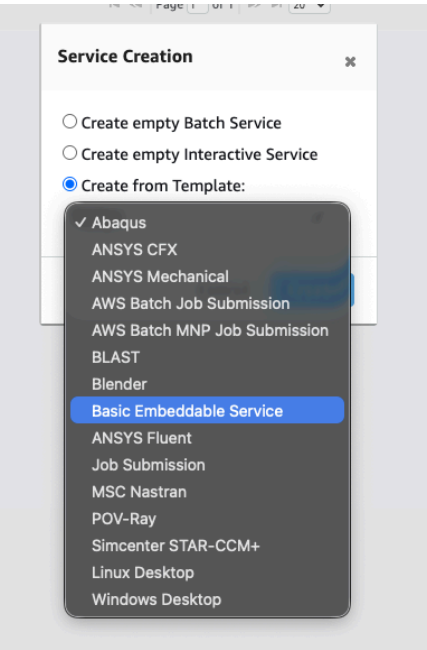


When you publish the service, you can restrict the view to individual users or user groups. You can also specify the Folder in which the service should appear.

Embeddable Services

Embeddable Services offer dynamic information in EF services. Information retrieved “on-the-fly” from the backend can be displayed in the service frontend e.g. querying a database for projects the user is assigned to, retrieving versions information of a software, getting load information of a queue or a cluster, or customized values.

To build an embeddable service in the Applications portal first create a new embeddable service starting from the available template:

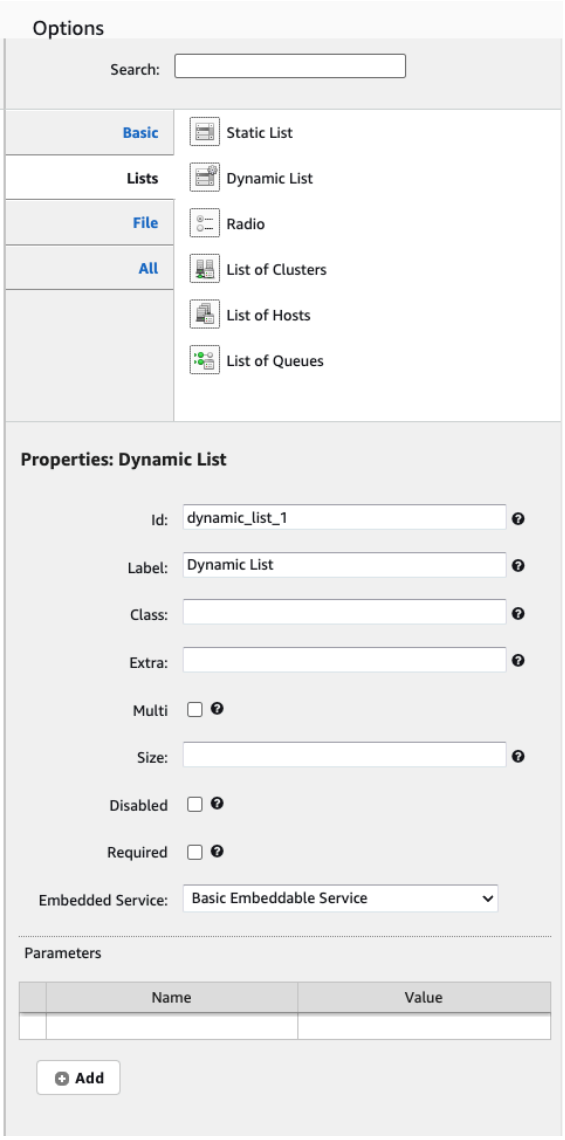


Then you can modify the action script of the new service to contain the action you want to perform, accordingly to your example should be something like:

```
. /etc/profile
module -t avail ansys |& grep -q ansys
if [ $? -eq 0 ]; then
ml -l avail ansys |& awk '/ansys/ { n = split($1,a,"/");
printf "<ef:option label=\"%s\"",a[n];
if ( $2 == "default" ) { printf " selected=\"YES\"" };
print ">" $1 "</ef:option>" }'
else
printf                    '<ef:option                    label="19.4"
selected="true">ansys/19.4</ef:option>'
fi
```

Then save the service and Publish it. Is required to publish it to make it available as embeddable services in other services. It is not visible to the users, it's a hidden service.

In the service where you want to use this embed service, you have to add a Dynamic List option. In the details of the dynamic list you can select the embeddable service you want (you can also pass specific parameters if you want):


Options
Search:
Basic
Lists
File
All
Static List
Dynamic List
Radio
List of Clusters
List of Hosts
List of Queues
Properties: Dynamic List
Id:
dynamic_list_1
Label:
Dynamic List
Class:
Extra:
Multi
Size:
Disabled
Required
Embedded Service:
Basic Embeddable Service
Parameters
Name
Value
Add

Security

Topics

Authentication framework

The EF Portal authentication framework supports a variety of different authentication methods. You can easily write your own custom mechanism to authenticate users when standard methods don't meet your specific requirements.

Standard EF Portal authentication authorities

When you install EF Portal, the following built-in authentication mechanisms are available:

- Pluggable authentication modules (PAM)
- Lightweight Delivery Access Protocol (LDAP)
- HTTP authentication
- Active Directory (AD) authentication
- Certificate-based authentication

Default authentication authority

When you install EF Portal, you choose what authentication method you want to be the default authentication method that's used to access services. You can change this setting later on. To change it, change the `EF_DEFAULT_AUTHORITY` property setting in the `server.conf` config file. For example, to set the default authentication method to PAM, change the relevant line to the following: `EF_DEFAULT_AUTHORITY=pam`.

The default authentication method that you set when you installed EF Portal is used by all the services that have the ef:agent's *authority* attribute set to `${EF_DEFAULT_AUTHORITY}`. This is shown in the following example.

```
<ef:agent xmlns:ef="http://www.enginframe.com/2000/EnginFrame"
 id="tutorial"
 authority="${EF_DEFAULT_AUTHORITY}">
```

You can set the authentication authority at the ef:service level to override the default authentication method that you chose when you installed EF Portal. The default authentication authority that you chose is defined in the root ef:agent tag.

User mapping

Users can log in to the EF Portal portal with a username that's different from that of their account in the underlying computing environment, when you configure EF Portal *user mapping*. User mapping works by mapping usernames provided at login time to usernames in the underlying operating system.

For example, the user *John Smith* can log into the EF Portal Portal using by entering John Smith and submit a job that's run as the user jsmith on the underlying Unix computing environment.

In some cases, user mapping can also be used to map different users of the portal to the same system account, or to map the same user to different accounts on separate systems.

Mapping root Account
You can't map users to the root account.

All of the authentication modules that are built-in and available when you install EF Portal support user mapping.

To enable user mapping, complete the following steps:

1. In the `$EF_TOP/conf/plugins/<authority>/ef.auth.conf` config file, set the `EFAUTH_USERMAPPING` parameter to true.
2. In the `EF_ROOT/plugins/<authority>/bin` directory of the authentication module, add a script that's named ef.user.mapping.
3. Set the ownership of the `ef.user.mapping` file to `root:root` and its permissions to 755 (`rwxr xr-x`).

The `ef.user.mapping` script must output the mapped username for the user that's being authenticated.

In the following example, ef.user.mapping maps all the portal users to the unique jsmith user.

```
#!/bin/sh

echo "jsmith"
```

In the following example, user mapping involves reading a file where each line specifies a mapping using the `Login Name=username` format.

For example, a file that's named `EF_ROOT/plugins/<authority>/conf/user.mapping` contains the following.

```
# simple mapping file
#
# Syntax: loginname=unixaccount
#

Lucy Johnson=ljohnson
John Smith=jsmith
```

In this example, the `ef.user.mapping` script attempts to match the name that's provided by the user during login with the ones that are present on the left-hand side of the equal sign in the user.mapping file. Then, as shown in the following output, it maps it to the corresponding account name on the right hand side of the equal sign.

```
#!/bin/sh

# read login name from command line
_loginname="$1"

# mapping file
_mappingfile=`dirname "$0"`"/../conf/user.mapping"
```

```
# transform login name so it can be used by sed in a safe way
_happysed=`echo "${_loginname}" | sed \
 -e 's#\\\\#\\\\\\\\#g' \
 -e 's#/#\\\\/#g' \
 -e 's/\\./\\\\./g' \
 -e 's/\\[/\\\\[/g'
`
# extract mapped unix account
_mapping=`sed -n 's/^'"${_happysed}"'=\(.*\)$/\1/p' "${_mappingfile}"`

# check with case insensitive flag if necessary
if [ -z "${_mapping}" ] ; then
       _mapping=`sed       -n       's/^'"${_happysed}"'=\(.*\)$/\1/pi'
"${_mappingfile}"` fi

# print first result
if [ -n "${_mapping}" ] ; then
 echo "${_mapping}" | sed 'q'
else
 exit 1
fi
```

With user mapping, an administrator can log in as an emulated user that's experiencing issues in a running production environment for testing purposes. This can be accomplished without compromising other accounts. With the following simple example ef.user.mapping file, a user named user1 takes on the identity of user2 without impacting other user accounts.

```
#!/bin/bash
if [["$1"==user1]] ; then
 echo "user2"
else
 echo "$1"
fi
```

Configuring the EF Portal authentication authorities

Besides the user mapping feature that's described in the previous section, the authentication authorities that are provided when you install EF Portal have additional configuration parameters that can be changed to tailor the authentication process to your environment.

Topics

PAM authentication

The pluggable authentication modules (PAM) authority authenticates a user using the PAM method of the Operating System.

The PAM_SERVICE parameter in the `$EF_TOP/conf/plugins/pam/ef.auth.conf` file on your Agent host specifies which PAM service is used for the authentication.

LDAP authentication

This authority authenticates users querying a LDAP database. You can configure the settings in the `$EF_TOP/conf/plugins/ldap/ef.auth.conf` config file on your Agent host. You can specify the location of the LDAP server to use and customize database access.

The following parameters are available:

- `LDAP_LDAPSEARCH`: the absolute path to the **ldapsearch** executable
- `LDAP_SERVER`: LDAP Server name or IP address
- `LDAP_PORT`: LDAP Server port
- `LDAP_BASE`: the base DN (Distinguished Name) that's used for the search operation in the LDAP database

Active Directory authentication

This authentication authority authenticates users querying an Active Directory database. You can configure the settings in the `$EF_TOP/conf/plugins/activedirectory/ef.auth.conf` config file on your Agent host. You can specify the location of the Active Directory server to use and customize database access.

The following parameters are available:

- `AD_LDAPSEARCH`: the absolute path to the **ldapsearch** executable
- `AD_SERVER`: LDAP Server name or IP address
- `AD_PORT`: LDAP Server port
- `AD_BASE`: the base DN (Distinguished Name) for the search operation in the Active Directory database
- `AD_BINDAS`: the user that has permissions to bind to an Active Directory Server for queries
- `AD_BINDPWD`: the password for a user binding to an Active Directory Server

Using signed certificates with EF Portal

HTTP authentication is different from the other EF Portal authentication methods: HTTP authentication is accomplished by the Web Server.

After the user is authenticated by the Web Server, the EF Portal HTTP authentication authority allows you to perform some initialization steps. Specifically, it allows you to use the EF Portal user mapping feature that was mentioned in the User mapping section. You can use this feature to map the EF Portal Portal users to the underlying operating system usernames.

Tip
When using HTTP authentication, in the case that you need to configure a user mapping that retrieves information from an LDAP Server, we recommend that you use the openldap software.

Certificate authentication

The certificate authority (CA) relies on X.509 certificates to encrypt the channel, check the client's identity, and authenticate users.

The EF Portal web server must be configured to use X.509 certificates for HTTPS channel encryption and client authentication. EF Portal uses a certificate authority to authenticate a user. You can configure either the Apache Tomcat® web server provided with EF Portal, or an external web server such as Apache Web Server connected to NI SP EF Portal Tomcat® with AJP connector to use certificates. For more information, see Configuring HTTPS.

For a client to authenticate to EF Portal, the client must provide a valid certificate the web server recognizes and trusts, and a username to be used by EF Portal to log in with.

EF Portal can be configured to retrieve the username from the first Common Name (CN) field of the Distinguished Name (DN) certificate property or from the HTTP request parameter named `_username`.

`authorization.certificate.userCertificate` is that parameter that's used to configure where EF Portal looks for the client username. It's in the server.conf config file. If this parameter is set true, the username is retrieved from the CN field of the DN in the client certificate. If this parameter is set to false, the username is retrieved from the client HTTP authentication request.

With the Certificate authority, like other authentication authorities, it's possible to use the user mapping feature that was mentioned in the User mapping section. This allows EF Portal users to be mapped to underlying operating system usernames.

Custom authentication authority

If none of the standard mechanisms included in EF Portal meet your specific requirements, you can create your own authentication algorithm.

The process of writing a custom EF Portal authentication module involves the development of the following components:

- The `EF_ROOT/plugins/<authority>/etc/<authority>.login` file. This file is used to specify the fields the users enter for authentication. Afterwards, the field values are passed to the following authentication module at login time.
- The `EF_ROOT/plugins/<authority>/bin/ef.auth` authentication module script. It receives the authentication field values that the user entered as input and completes the authentication.

After the new authority is ready, it can be used in the authority attribute for ef:agent and ef:service tags in the service definition file (SDF).

Warning
If one of the authentication authorities that's included in EF Portal doesn't quite meet your specific requirements, *do not modify EF Portal system files*. Rather, create your own authority. You can use the authentication authorities that are provided with EF Portal as a starting point.
It's important to note that modifying one or more of the EF Portal system files might corrupt the system. We strongly recommend that you do not modify EF Portal system files. Only modify EF Portal system files at your own risk. NI SP and its partners do not respond for EF Portal unresponsiveness if a system file has been modified. Understand also that a future EF Portal update might override your modifications without warning.

The *<authority>*.login file

This XML file defines the authentication parameters that are required by the authentication script to check user credentials.

This file specifies the information (including the username, token, and password) and prompts for the logon pages that are associated with the authentication module.

The login file must match the authority name, that is, the name that will be used in the EF Portal SDF. It must also be located under the `EF_ROOT/plugins/<authority>/etc` directory. For security reasons, the file ownership must be set to root:root and its permissions must be 644 (`rw-r--r--`).

For example, the login file for authority *PAM* must be: `EF_ROOT/plugins/pam/etc/pam.login` The `<authority>.login` file has the following

structure:

```
<ef:login title="login_form_title"
 xmlns:ef="http://www.enginframe.com/2000/EnginFrame">;
 <ef:signature label="login_field_label"
 type="text|password"
 id="authentication_parameter_name" />;
 <!-- [<ef:signature ... />] -->
</ef:login>
```

The following example is taken from the *PAM* authority:

```
<ef:login title="Login to EF Portal">
 <ef:signature label="Username: "
 type="text"
 id="_username"/>
 <ef:signature label="Password: "
 type="password"
 id="_password"/>
</ef:login>
```

The `ef:login` entry corresponds to an authentication HTML page. Referring to the example, the HTML login page asks users to enter a text token (the username) and a password.

Note
We strongly recommend that you use the authentication parameters `<ef:signature>` with `id=_username` and `id=_password` (the `<ef:signature>` tags of the pam.login example). Otherwise, you must include `<ef:user-mapping>` in the `<authority>.login` file to provide the outputs that EF Portal needs to retrieve the username.

The ef.auth File

The script ef.auth actually implements the authentication procedure.

It must reside under the directory `EF_ROOT/plugins/<authority>/bin`. The *<authority>* must match the authority name that will be used in the EF Portal SDF. For security reasons, the file ownership must be set to `root:root` and its permissions must be set to 755 (`rwxr-xr-x`).

For example, the authentication script for the LDAP authority is `EF_ROOT/plugins/ldap/bin/ef.auth`.

The authentication script receives as input the login form parameters values that the user entered. Such values, separated by '`\0`' character (ASCII code 0), are directly passed to the ef.auth script in the same order that they're defined in the `<authority>.login` file.

Consider the *PAM* authentication authority as an example. When a user that's named demo with a password secret logs into EF Portal, the string `demo\0secret\0` is passed to the `ef.auth` script.

The authentication script checks the credentials passed as input and writes the response to the standard output. If the credential check was successful, the response is as follows:

```
<?xml version="1.0"?>
<ef:auth xmlns:ef="http://www.enginframe.com/2000/EnginFrame">

 <ef:result>
 <ef:grant />
 </ef:result>

</ef:auth>
```

If it isn't successful, the response is as follows:

```
<?xml version="1.0"?>
<ef:auth xmlns:ef="http://www.enginframe.com/2000/EnginFrame">

 <ef:result>
 <ef:deny />
 </ef:result>

 <ef:error> <!-- Not mandatory -->
 <ef:message>error_message</ef:message>
 </ef:error>

</ef:auth>
```

If the credential check was successful and you have configured user mapping for your custom authentication authority, the ef:user-mapping tag is included in the `ef.auth` script output:

```
<?xml version="1.0"?>
<ef:auth xmlns:ef="http://www.enginframe.com/2000/EnginFrame">

 <ef:result>
 <ef:grant />
 <ef:user-mapping name="target_username"/>
 </ef:result>

</ef:auth>
```

target_username is the username that's used on the underlying operating system.

We recommend that you move the user mapping logic to a separate script that can be changed without editing ef.auth itself. The authentication modules that are built into EF Portal follow the convention of using the `EF_ROOT/plugins/<authority>/bin/ef.user.mapping` script. For more information, see User mapping.

If you move the user mapping logic to a separate script, your `ef.auth` script has the following structure:

```
[Get credentials]

[Verify credentials]

[If User is authenticated]
 ACTUAL_USERID=[Custom User Mapping procedure]

 [Emit]
 <?xml version="1.0"?>
     <ef:auth     xmlns:ef="http://www.enginframe.com/2000/EnginFrame">
<ef:grant/>
 <ef:user-mapping name="$ACTUAL_USERID"/>
 </ef:auth>
 [end]
[end]
```

Configurable logout behavior

The configuration file `EF_ROOT/plugins/ef/conf/ef.logout.conf`, containing the new parameter `EF_REDIRECT_ON_LOGOUT`, offers to configure the behavior of the portal after logout.

If set to true (it is the default), users will be automatically redirected to the default service of the plugin being used, which usually triggers the login page. If set to false, a standard logout page will be shown instead.

Setting to false is particularly useful when the 'certificate' authority is used, since redirecting to a service would automatically re-login the user.

You can apply the conf at global level using `EF_CONF_ROOT/plugins/ef/ef.logout.conf` (to be created if it does not exist).

Authorization system

You can use the EF Portal Authorization System to control which *users* can access the EF Portal *resources* based on policies that you set. Through these policies, you can grant or deny a user access to specific resources or to do specific operations.

In the EF Portal Authorization System, *users* and *groups* are called Actors, *resources* are divided into EF Portal service definition files (SDF) folders, services, service options, service actions and service output, and the *policies* define the permissions that are specified using access control lists (ACL).

The authorization framework defines *who* can do *what* on *which resources*. By configuring the authorization system, you can give different views of the EF Portal to different users and user groups.

Configuring authorization

EF Portal authorization settings are specified in the following configuration files. The ACLs and Actors that are defined in all of them are merged. If there are multiple definitions of the same ACLs or Actors, the following file priority is used to resolve conflicts, from highest to lowest priority.

- `$EF_TOP/conf/enginframe/authorization.xconf` (highest priority)

- `$EF_TOP/<VERSION>/enginframe/conf/authorization.xconf` (medium priority)
- `$EF_TOP/conf/plugins/<plug-in>/authorization.xconf` (low priority)
- `$EF_TOP/<VERSION>/enginframe/plugins/<plug-in>/conf/authorization.xconf` (lowest priority)

Modifications to these files are automatically picked up by a running EF Portal Server, without requiring a restart.

The authorization.xconf file is an XML file consisting of two sections:

- An *Actors* section that's introduced with the tag `<ef:acl-actor-list>`
- An *Access Control Lists* section that starts with the tag `<ef:acl-list>`

```
<ef:authorization>
 <!-- EF Portal Authorization Actors section -->
 <ef:acl-actor-list>
 ...
 </ef:acl-actor-list>
 <!-- EF Portal Authorization ACL section -->
 <ef:acl-list>
 ...
 </ef:acl-list>
</ef:authorization>
```

Topics

- Defining Actors
- Defining access control lists
- Condition based ACL

Defining Actors

Actors are entities that can perform actions on resources. In the EF Portal infrastructure an Actor can be a user, a group of users, or a group of groups of users.

Because an Actor can be a single user or a group, we refer to each component using the term *member*. There are three ways to define an Actor:

- *efgroup Actor* is an explicit list of members. It can contain one or more users or even other Actors already defined.
- *osgroup Actor* is a list of members that includes the users that belong to the Operating System group that also has the same ID as the Actors associated with it.
- *user Actor* is an actor for a single EF Portal user. It exists just for "renaming" purposes because usually there's no need to wrap an EF Portal user ID in an Actor.

The XML syntax that's used to define Actors inside authorization.xconf config file is as follows:

```
<ef:acl-actor id="unique_id” type="efgroup|osgroup">
```

An Actor of type `efgroup` must include members defined with the following:

```
 <ef:acl-member type="efuser|acl-actor">...</ef:acl-member>
```

Note
When using a member of the acl-actor type, the Actor must be defined in the authorization.xconf

config file.

The following is an Actors section example:

```
...
<!-- EF Portal Authorization Actors section -->
<ef:acl-actor-list>
 <!-- Actor made up of two simple users and two Actors -->
 <ef:acl-actor id="nice" type="efgroup">
 <ef:info>NI SP people</ef:info>
 <ef:acl-member type="efuser">enrico</ef:acl-member>
 <ef:acl-member type="efuser">francesco</ef:acl-member>
 <ef:acl-member type="acl-actor">
 developers
 </ef:acl-member>
 <ef:acl-member type="acl-actor">efadmin</ef:acl-member>
 </ef:acl-actor>
 <ef:acl-actor id="developers" type="efgroup">
 <ef:info>EF Portal Developers</ef:info>
 <ef:acl-member type="efuser">antonio</ef:acl-member>
 <ef:acl-member type="efuser">mauri</ef:acl-member>
 <ef:acl-member type="acl-actor">goldrake</ef:acl-member>
 </ef:acl-actor>
 <!--
 Member of this Actor are dynamically loaded from the
 Operating system group “efadmin” using the script:
 NISP_ROOT/efportal/plugins/myplugin/bin/ef.load.users
 -->
 <ef:acl-actor id="efadmin" type="osgroup" plugin="myplugin"/>
</ef:acl-actor-list>
...
```

Defining access control lists

Access control lists (ACL) define policies that are applied to Actors when they try to perform actions on resources.

The purpose of an access control list (ACL) is to grant or deny permissions on actions that an EF Portal Actor can perform without reference to a specific resource.

An ACL consists of three main parts. The first section is used to "bias" the ACL towards the allow or the deny directive. The following two parts define the allow and deny directives where Actors are bound with the actions they can or cannot perform.

The ACL structure explanation follows

- *ACL priority*, defines if the allow or deny directive has priority for this ACL:
    - allow priority: By default, access is allowed. The deny directives are evaluated before the allow ones. Any Actor that doesn't match a deny directive or matches an allow directive is allowed access to the resource.
    - deny priority: By default, access is denied. The allow directives are evaluated before the deny ones. Any Actor that doesn't match an allow directive or matches a deny directive is denied access to the resource.
- *ACL allow*, contains the allow directives, a list of Actors where a set of actions specifies the operations that the Actor can actually perform on a generic resource guarded by this ACL.
- *ACL deny*, contains the deny directives, a list of Actors where a set of actions specifies the

operations that the Actor can't perform on a generic resource guarded by this ACL.

The definition of an ACL adheres to this XML structure:

```
...
 <ef:acl id="unique_id”>
 <ef:acl-priority>allow | deny</ef:acl-priority>
 <ef:acl-allow>
 <ef:actor id=”actor_id”>
 <ef:action-list>
 <ef:read/>
 <ef:execute/>
 ...
 </ef:action-list>
 </ef:actor>
 </ef:acl-allow>
 <ef:acl-deny>
 <ef:actor id=”actor_id”>
 <ef:action-list>
 <ef:read/>
 <ef:execute/>
 ...
 </ef:action-list>
 </ef:actor>
 </ef:acl-deny>
 </ef:acl>
...
```

The id attribute of an ef:actor can refer to a predefined `ef:acl-actor` or directly to an EF Portal user ID.

There are four kinds of actions EF Portal can accept. The action meaning and the consequent EF Portal behavior depends on the type of the resource that the ACL is applied to. The four kinds of actions are the following:

- `<ef:read/>`
- `<ef:write/>`
- `<ef:execute/>`
- `<ef:delete/>`

Here an example of an ACL definition follows:

```
...
 <ef:acl-list>


 ...
 <ef:acl id="priv-exec">
 <ef:info>Privileged permissions for Admins</ef:info>
 <ef:acl-priority>deny</ef:acl-priority>
 <ef:acl-allow>
 <ef:actor id="efadmin">
 <ef:action-list>
```

```
 <ef:read/>
 <ef:write/>
 <ef:execute/>
 <ef:delete/>
 </ef:action-list>
 </ef:actor>
 </ef:acl-allow>
 </ef:acl>
 ...
 </ef:acl-list>
 ...
```

Condition based ACL

An ACL can include some extra conditions on granting access to a resource.

These conditions can take into account the value of session variables, system properties, and xpath expressions and be combined using the logical operators: or, and, not, and equals.

This is illustrated in the following example.

```
...
<ef:acl-list>
 ...
 <ef:acl id="project-acme">
 <ef:info>
 Privileged permissions for Project ACME
 </ef:info>
 <ef:acl-priority>deny</ef:acl-priority>
 <ef:acl-allow>
 <ef:actor id="company-users">
 <ef:condition>
 <ef:or>
 <ef:and>
 <ef:equals type=”session” id=”project” value=”acme”
 casesensitive=”true”/>
 <ef:equals type=”session” id=”${project}_responsible” value=”true”
casesensitive=”false”/>
 </ef:and>
 <ef:and>
 <ef:equals type=”session”
 id=”administrator”
 value=”true”
 casesensitive=”false”/>
 <ef:not>
 <ef:equals type=”property”
 id=”${EF_USER}”
 value=”jack”
 casesensitive=”true”/>
 </ef:not>
 </ef:and>
 </ef:or>
 </ef:condition>
 <ef:action-list>
 <ef:read/>
 <ef:write/>
```

```
 <ef:execute/>
 <ef:delete/>
 </ef:action-list>
 </ef:actor>
 </ef:acl-allow>
 </ef:acl>
 ...
</ef:acl-list>
```

A user can access resources guarded by the "project-acme" ACL only if one of the following conditions is true:

- The user belongs to "company-users", the session variable "project" is present and its value is "acme", and the session variable "acme_responsible" is set to "true" independently from the case of letters.
- The session variable "administrator" is “true” independently from the case of letters, and it isn't named "jack".

The `ef:and` and `ef:or` tags are condition containers. `ef:and` evaluates to true when all of the included conditions evaluate to true. ef:or evaluates to true when at least one of its conditions evaluates to true.

The `ef:not` might contain only one condition and it negates the result of its evaluation.

The ef:equals tag checks two arguments for equality. The arguments to check are defined by the type and by the id attributes. The case sensitive attribute specifies if the letter case matching is taken into account.

The type can refer to three different kinds of values:

Session variables

The value to be checked is a session variable with the specified id.

The following is the session variable from the preceding example.

```
<ef:equals type=”session”
 id=”administrator”
 value=”true”
 casesensitive=”false”/>
```

EF Portal checks if a session variable named administrator is defined and its value is true.

System properties

The value to be checked is a system property with the specified id. EF Portal system properties are loaded by the JVM, passed to the JVM using the command line (for example, **`-Dname=value`**), and loaded from the EF Portal configuration files.

```
<ef:equals type=”property”
 id=”${EF_USER}”
 value=”mary”
 casesensitive=”true”/>
```

EF Portal checks if the system property named `${EF_USER}` has the value mary.

XPath expressions

The value to be checked is the one extracted from the current DOM by the XPath expression specified in the id attribute.

```
<ef:equals type="xpath"
 id="starts-with(//ef:profile/ef:user/., 'br')"
 value="true"
 casesensitive="true"/>
```

EF Portal checks if the HTML DOM element `//ef:profile/ef:user/text()` starts with `br`.

Configuring HTTPS

HTTPS, Hypertext Transfer Protocol over Secure Socket Layer, is a URI scheme used for secure HTTP connections.

HTTPS is a protocol that adds a layer of encryption to security-sensitive communication such as payment transactions and corporate logons.

During the installation process, EF Portal installer has an option for Apache Tomcat® to use HTTPS instead of HTTP. By default, EF Portal installs with the option to use HTTP protocol in the Apache Tomcat® web server.

If you choose the HTTPS option, the installer automatically creates self-signed certificates under the `$EF_TOP/conf/tomcat/conf/certs` directory. It also configures the Apache Tomcat® connector to use HTTPS.

Using signed certificates with EF Portal

If self-signed certificates aren't suitable for your needs or you already have valid certificates setup for an HTTPS web server, we recommend that you refer to the Apache Tomcat® official documentation when configuring your web server to use your certificates.

Apache Tomcat® provides different setup procedures depending on the specific format of the available certificate.

For example, if using a PEM private key and PEM certificate, the key and certificate must be converted before they are handled by Java™ keytool and keystores. This is the best practice that's recommended in the Apache Tomcat® documentation.

First, convert PEM key and certificate into PKCS12 format:

```
$ openssl pkcs12 -export -in <your_CA_signed_PEM_cert>
 -inkey <your_PEM_private.key> -out <your_certificate_name>.p12
 -name tomcat -chain -CAFile <your_root_CA_certificate>
```

Next, import the newly created PKCS12 certificate into a Java™ keystore file:

```
$ $JAVA_HOME/bin/keytool -importkeystore -deststorepass <password>
 -destkeypass <password> -destkeystore tomcat.keystore
 -srckeystore <exported_private_key_and_cert.p12> -srcstoretype PKCS12
```

```
 -srcstorepass <password> -alias tomcat
```

If your certificate authority (CA) has intermediate certificates, import them into the keystore file. It's likely that your CA provides instructions on how to do this and how to name certificates. For example, if a CA intermediate certificate is already in a format that's supported by Java™ keytool, you can import it this way:

```
$ $JAVA_HOME/bin/keytool -import -alias intermed -keystore
tomcat.keystore -trustcacerts -file gd_intermediate.crt
```

You might also need to import the root CA certificate into the keystore if it doesn't come from one of the CAs whose root certificates are pre-configured in the Java™ system.

For more information about how to use the keytool and openssl commands, in the Oracle Java SE Documentation.

Configuring SSL/TLS for Hazelcast

Transport Layer Security (TLS) is a cryptographic protocol designed to provide communications security over a computer network.

With Hazelcast you can encrypt socket level communication between Hazelcast members and between Hazelcast clients and members, for end to end encryption.

EF Portal ships with Hazelcast Open Source which doesn't support the security suite that includes SSL/TLS asymmetric encryption.

Setup SSL/TLS communication with Hazelcast Enterprise

1. To learn how to set up SSL/TLS communication with Hazelcast Enterprise, you need:
    a. A valid Hazelcast Enterprise license.
    b. The Hazelcast Enterprise jar (current version 5.3.5).
2. To begin, replace `${EF_ROOT}/WEBAPP/WEB-INF/lib/hazelcast-5.3.5.jar` with the downloaded Hazelcast Enterprise jar.
3. After the jar is replaced, update the Hazelcast configuration file `${EF_ROOT}/conf/hazelcast.xml` with the license key that you already obtained by following this guide.
4. 4. Implement com.hazelcast.nio.ssl.SSLContextFactory and configure the SSL section in the network configuration. Hazelcast provides a default SSLContextFactory, `com.hazelcast.nio.ssl.BasicSSLContextFactory`, that uses the keystore to initialize SSLContext. An example configuration for TLS/SSL follows:

```
<hazelcast>
...
<network>
 ...
 <ssl enabled="true">
 <factory-class-name>
 com.hazelcast.nio.ssl.BasicSSLContextFactory
 </factory-class-name>
 <properties>
 <property name="protocol">TLSv1.2</property>
  <property name="mutualAuthentication">REQUIRED</property>    <property
name="keyStore">/efs/KeyStore.jks</property>
 <property name="keyStorePassword">passphrase</property>
 <property name="keyStoreType">JKS</property>
```

```
 <property name="trustStore">/efs/truststore.jks</property>   <property
name="trustStorePassword">passphrase</property>                <property
name="trustStoreType">JKS</property>
 </properties>
 </ssl>
</network>
...
</hazelcast>
```

Property descriptions:

- **keyStore**: Path of your keystore file.
- **keyStorePassword**: Password to access the key from your keystore file.
- **keyManagerAlgorithm**: Name of the algorithm based on the provided authentication keys.
- **keyStoreType**: Type of the keystore. Its default value is JKS. Another commonly used type is the PKCS12. Available keystore/truststore types depend on your operating system and Java runtime.
- **trustStore**: Path of your truststore file. The truststore file is a keystore file that contains a collection of certificates trusted by your application.
- **trustStorePassword**: Password to unlock the truststore file.
- **trustStoreType**: Type of the truststore. Its default value is JKS. Another commonly used type is the PKCS12. Available keystore/truststore types depend on your operating system and Java runtime.
- **mutualAuthentication**: Mutual authentication configuration. It's empty by default which means the client side of the connection is not authenticated. Available values are:
    - REQUIRED - server forces usage of a trusted client certificate.
    - OPTIONAL - server asks for a client certificate, but doesn't require it.

For more information, see the Hazelcast documentation.

5. Follow the next example to properly configure a Keystore and a Truststore that leverages both keytool and openssl.

```
$ keytool cd /efs
keytool -genkey -alias bmc -keyalg RSA -keystore KeyStore.jks -keysize
2048 openssl req -new -x509 -keyout ca-key -out ca-cert
keytool -keystore KeyStore.jks -alias bmc -certreq -file cert-file
openssl x509 -req -CA ca-cert -CAkey ca-key -in cert-file -out
cert-signed -days 365 - CAcreateserial -passin pass:passphrase
keytool -keystore KeyStore.jks -alias CARoot -import -file ca-cert
keytool -keystore KeyStore.jks -alias bmc -import -file cert-signed
keytool -keystore truststore.jks -alias bmc -import -file ca-cert
```

With this example, you create a custom CA with openssl that's used to sign the Hazelcast certificate. We recommend that you use a publicly available CA for production and reserve the use of custom CA certificates for testing purposes.

6. Restart EF Portal to leverage SSL/TLS for Hazelcast members and clients communications.

Document History

The following table describes the major updates and new features of the respective release.

Date: Apr 27, 2026 - Version 2026.0

NEW FEATURES:

* Introduced a new Web Configuration Editor service that enables administrators to modify configuration parameters and files directly from the portal, without accessing the underlying file system. The service also provides built-in validation of the configured settings.
* Introduced a real-time notification system that delivers in-portal alerts for key events, such as job status changes and system status updates. Administrators can also send custom notifications to all users in a role or to specific users through a dedicated UI.
* File Manager user interface now enables users to add, modify, remove, and sort their places. This allows them to easily organize their data into folders, projects, local and remote data stores.
* Added a Log File Viewer to the Operational Dashboard (Troubleshooting section) with themed viewer integration, in-file search with navigation, tail mode with auto-refresh, log level color coding, multi-host support, cross-file keyword search with CSV export.
* Operational Dashboard: Introduced a new service for navigating EF Portal release notes

ENHANCEMENTS:

* Technology Showcase: Added a comprehensive REST API demonstration to include practical use cases utilizing both EFPClient and cURL
* Technology Showcase: Added implementation examples for creating new notifications using both JavaScript and Bash scripts
* Operational Dashboard: The CPU Load and JVM Usage widgets now display data for all servers in an EF Portal Enterprise installation
* Slurm Plugin: Improved job status mappings when 'sacct' is used to interact with the scheduler
* Slurm Plugin: Added support for compact nodelist format
* Grid Plugin: Always use the host timezone offset for job submission to ensure the correct time is shown in the UI
* VDI: Display session ID in the Session Details page

* VDI: Link job ID in Session Details to the underlying Job Details page
* VDI: Do not display the Host link until the session job has started
* VDI: Added possibility to disable Zoom Preview in List Sessions page
* Applications / VDI: Added a link to the Operational Dashboard in the Admin View
* DCV SM Integration: Reduced the time-to-live of spoolers associated with broken sessions to minimize repetitive warning entries in the logs
* Service Editor: Preserve the scroll position of the options properties
* Service Editor: Show a description and a tooltip for each option widget
* UI: Show the pending reasons in the Jobs Details page
* UI: Display the cluster label in Host View when configured
* UI: Preserve the selected items in a list across refreshes
* UI: Introduced support for searching jobs by owner in the Jobs List
* UI: Disable the submit button after a service is submitted to prevent duplicate submissions
* UI: Each navigation menu item now exposes a stable CSS class (ef-nav-id-<service-id>) so that customers can target individual menu entries with custom CSS
* Applications / VDI: Added possibility to customize refresh interval in the List Services and List Interactive Services pages
* File Manager: Compute place name if description is missing in ef.places
* File Manager: Added configurable parameter to permit to follow symlinks outside the virtual root (vroot)
* File Manager / RFB: Display base place path in the location bar
* File Manager: Enable up arrow leading to parent folder
* File Manager: Avoid compression when downloading a single file
* File Manager: Show spinning icons and disable the related button when downloading or compressing multiple files
* Remote File Browsing: Save the last visited folder when it is loaded from a profile
* Authentication: Added configurable parameters for AD/LDAP filters
* Authentication: Added possibility to configure the URL to redirect after a logout
* Core: Improved <ef:redirect> tag with the newWindow attribute to open the target page in a new window or tab in the browser
* REST API: Added new service to list available tokens for the logged-in user
* REST API: Administration endpoints are now accessible by Applications and VDI administrators
* Interactive: Improved creation of spoolers for interactive sessions to prevent the generation of inconsistent spoolers in case of errors
* Improved error reporting for Ajax service calls
* Improved independent control of EF Portal daemons via the 'enginframe' startup script
* LSF Plugin: Allow GPU info parser to handle non-numeric UT/MUT values
* LSF Plugin: Improved OS compatibility by initializing the execution environment through a configurable shell
* Job Cache: Improved handling of duplicate job entries when cluster IDs are missing or inconsistent
* Job Cache: Introduced configurable batch processing for job updates to minimize the risk of deadlocks

* PBS Plugin: Improve parsing of grid:execution-host GridML tag
* PBS Plugin: Added support for submitting interactive sessions on OpenPBS 23.06
* Hosts Info: Quick Commands are now shown also for Applications and VDI Administrators
* Operational Dashboard: Extended Cleanup Interactive Sessions service to be able to clean up sessions in any state.
* Improved Spoolers cleanup process to avoid repetitive execution in case of unexpected errors

DISCONTINUED INTEGRATIONS:
* Discontinued support for Java 8. Use Java 11 before upgrading EF Portal

CHANGES:
* Web Terminal is now disabled by default. Users who previously relied on it can easily re-enable it through the Web Configuration Editor
* DCV SM Integration: Session Recovery is now disabled by default
* Slurm Plugin: The "dcv2" feature is no longer required for interactive sessions
* Slurm Plugin: Added optional SLURM_INTERACTIVE_FEATURE parameter to specify custom constraints for interactive sessions
* UI: Added configurable auto-refresh for the Jobs List and Spoolers List tables
* UI: Changed the default Jobs List ordering to descending by submission time
* Upgraded Apache Tomcat to version 9.0.117
* Upgraded Apache Derby to version 10.15.2.0
* Upgraded Apache Log4j to version 2.25.4
* Upgraded Apache Commons FileUpload to version 1.6.0
* Upgraded Apache Commons IO to 2.19.0
* Upgraded Apache Commons Lang to version 3.18.0
* Upgraded Apache Commons Logging to version 1.3.5
* Upgraded Apache Commons BeanUtils to version 1.11.0
* Upgraded Apache Commons Net to version 3.13.0
* Upgraded AWS SDK for Java to version 2.41.34
* Upgraded C3P0 to version 0.12.0
* Upgraded Hazelcast to version 5.3.8
* Upgraded Jackson JSON library to version 2.21
* Upgraded SLF4J to version 2.0.16
* Upgraded Swagger UI to version 5.32.0
* Upgraded SnakeYAML to version 2.6
* Upgraded Swagger JAXRS2 to version 2.2.45
* Upgraded Kotlin libraries to version 1.7.22
* Upgraded Node.js to version 22.22.2
* Upgraded Node.js modules used by Web Terminal and Proxy
* Removed Spring Framework dependency

FIXES:
* Slurm Plugin: Use SLURM_CLUSTER_LABEL when it is configured
* Torque Plugin: Fix host information parsing
* Service Profile does not store Anti CSRF Token information

Date: Dec 17, 2025 - Version 2025.3

NEW FEATURES:

* Added Web Proxy Server to enable access to internal cluster nodes and services directly from EF Portal, enabling access to e.g. Jupyter notebooks or Web Terminals not directly accessible from outside of the cluster
* Operational Dashboard: Added new service to cleanup Interactive Sessions

ENHANCEMENTS:

* VDI: Show DCV error message in the session log if DCV session start fails e.g. related to missing license
* VDI: Added automatic closure of interactive sessions if the underlying job or delegate session remains in an unknown state for a configurable period
* Slurm Plugin: Improved job status mappings with additional states
* Slurm Plugin: Enhanced error reporting for failed interactive job submissions
* LSF Plugin: Show data for up to 8 GPUs per node in the Hosts view
* Improved automatic redirection to the login page when the web session expires
* REST API: Improved auditing and logging of REST calls
* REST API: The token creation service can be configured to output only the token value
* REST API: Improved endpoints documentation

CHANGES:

* Web Terminal now connects via HTTPS by default
* INTERACTIVE_* variables are set from session starting hook for other hooks to use
* Operational Dashboard: Job Cache monitor services have been moved into the Advanced Administration navigation folder
* Slurm Plugin: Refined parsing of host states to ensure nodes in a drain state are no treated as active
* File Manager: Updated utility libraries to ensure compatibility with /bin/sh shell
* VDI: INTERACTIVE_* variables are set from session starting hook for other hooks to use
* Auditing: Logout due to session expiration is now logged as a [LOGOUT] event

Date: Nov 13, 2025 - Version 2025.2

NEW FEATURES:

* Operational Dashboard: Added Administrator Dashboard to monitor the status of EF Portal in one comprehensive view
* Applications / VDI: Added functionality to directly configure and display a Message Of The Day (MOTD)
* VDI: Added the ability to pass additional requirements from pre-submit hook
* VDI: Added configurable timeouts for session hooks
* AWS Workspaces integration: Added sample pre-submit and closing hooks to manage AWS WorkSpaces instances and run DCV session including Office support
* Applications / VDI: Enabled configuration of services and folders on top of the default navigation menu

* Operational Dashboard: Added Job Cache administration services

ENHANCEMENTS:
* Remote File Browsing: Last visited folder is preserved across different RFBs in the same service page
* Remote File Browsing: Added the ability to configure a RFB to guide the last visited folder for all RFBs in the same service page
* Remote File Browsing: Added tooltip for full path visibility
* File Manager: Improved disk space management with automatic cleanup of download spoolers upon completion
* Operational Dashboard: Enabled access for HPC Workspace and Virtual Desktop administrators
* Operational Dashboard: Added quick link to the NI SP Support and Knowledge Base
* Operational Dashboard: Enhanced system statistics retrieval to reduce load on the EF Portal host
* File Manager: target address of a .url file can now be opened clicking on the filename
* DCV SM Integration: Added the ability to configure automatic session recovery
* DCV SM Integration: Added comprehensive support for requirements conditions and operators passed as submit options
* LSF Plugin: Improved compatibility to support a wider range of Linux environments and interpreters
* PBS Plugin: Optimized number of queries to PBS server
* PBS Plugin: Added support for Job IDs exceeding 30 characters
* PBS Plugin: Improved compatibility to support a wider range of Linux environments and interpreters
* Installer: Added umask detection and validation
* Added REST service in Admin SDF to retrieve CPU load and Memory usage
* Added new trigger to remove expired REST tokens from database
* Service Editor: Introduced checks for incomplete Service configuration, to prevent launch failures with unmodified default Services
* Service Editor: Improved validation rules and messaging for Max Number of Sessions parameter

CHANGES:
* VDI: Introduced a default timeout of 30 seconds for closing hook execution
* VDI: Execute hook scripts as separate processes instead of sourcing them
* VDI: Automatically switch to the Session folder when submitting a VDI job
* Changed default output limit for job scheduler XML output from 50MB to 300MB
* Installer: Added two-step confirmation when installing EF Agent on a separate host
* Installer: Technology Showcase and documentation are now installed by default
* Removed administration services from Technology Showcase portal
* Upgraded Apache Tomcat to version 9.0.111
* Upgraded rrd4j to version 3.9
* Upgraded Jackson JSON library to version 2.15.0
* Upgraded Apache Commons Text to version 1.14.0

BUGFIXES:

* VDI: Prevent duplicate execution of Interactive Session closing hooks upon session termination
* VDI: Prevent concurrent execution of Interactive Session starting hooks
* VDI: Resolved edge cases causing Linux sessions to hang in the Closing state
* Installer: Manage Web Terminal port configuration during updates
* Slurm Plugin: Properly display free memory in the hosts information page
* Service Editor: Fixed selectors reloading to prevent test run failures when switching between clusters
* Charts: Fixed chart rendering for series with all zero values
* Operational Dashboard: Fixed initialization of historical charts on first login

Date: July 17, 2025 - Version 2025.1

NEW FEATURES:

* Monitoring to customer owned StatsD/Grafana of EF Portal data such as jobs status, licenses status, logged users, CPU load, filesystem and memory usage
* Interactive Sessions pre-submit and closing hooks as extension points to support executing custom actions before the session is actually submitted and as soon as the session is closed
* Added WebTerminal to give administrators immediate access to remote hosts in the cluster from the Host View using a web-based terminal
* REST API endpoints for VDI management, such as show running sessions, get session information, connect to a session and close a session
* REST API automatically converts to JSON any EF Portal service output when it is in XML format, in addition to the original XML output

ENHANCEMENTS:

* File Manager: Improved Drag and Drop upload
* Service Editor: Improved locking of services
* Service Editor: Improved token management for Jupyter Notebook service
* Service Editor: Improved ACL management after editing a published service
* UI: Updated logic to prevent quick clicks on session start function
* Server: startup time reduced by about 40%
* Core: Number of other performance improvements
* Core: Removed legacy NEUTRO code and dependency
* Core: Replaced BSF/BSH with Groovy
* REST API: /monitor endpoints do not consume a REST license

CHANGES:

* Upgraded Apache Tomcat to version 9.0.107
* Upgraded Hazelcast to version 5.3.5
* Upgraded Quartz to version 2.4.0
* Upgraded Apache CXF to version 3.5.11
* Upgraded Apache Commons Lang to version 3.17.0
* Upgraded Apache Commons Text to version 1.13.1
* Upgraded Apache Commons IO to version 2.19.0

* Upgraded Apache Commons Logging to version 1.3.5
* Upgraded Apache Commons Codec to version 1.18.0
* Upgraded Apache Commons JEXL to version 2.1.1
* Upgraded Jackson to version 2.13.5
* Upgraded Google Guava to version 32.0.1-jre
* Upgraded Google Guava FailureAccess to version 1.0.3
* Upgraded Google Gson to version 2.13.0
* Upgraded JFreeChart to version 1.0.19
* Upgraded Yauaa to version 7.30.0
* Upgraded OWASP Java Encoder to version 1.3.1
* Upgraded HyperSQL Database to version 2.7.4
* Upgraded C3P0 to version 0.10.2
* Upgraded Mchange Commons Java to version 0.3.2

Date: January 17, 2025 - Version 2025.0

NEW FEATURES:

* Service Manager: added the ability to edit published services
* Core: Added auditing of logins and logouts into log file
* HPC Workspaces and Virtual Desktop: added CSS editor in Appearance Editor to fully customize the layout

ENHANCEMENTS:

* VDI: improved usability of the Connection Quick Access bar available for each session preview in the List Sessions compact view
* VDI: improved user experience when waiting to connect to a session in case either Web Client or DCV URI are used
* DCV SM Integration: added automatic recovery of running sessions available in the SM Broker not tracked by EF Portal
* File Manager: added preview of video files
* File Manager: added preview of glb and glTF 3D models
* File Manager: administrators can define custom places for all users using a global ef.places file
* File Manager: custom places defined by users outside their home directory must be allowlisted by administrators to be used
* File Manager: added the ability to define custom extensions for text and image mime types which will be recognized as such by the File Manager
* SLURM plugin: added new Starting and PoweredDown states, particularly useful when hosts are in the Cloud
* SLURM plugin: added Move Top action for pending jobs
* Settings: added DCV URI as default connection option. Improved settings pages usability
* Hosts view: GPU related columns are hidden if no GPU data is available
* UI: added configurable logout behavior
* Appearance Editor: improved layout and behavior
* Installer: optimal memory settings for Server and Agent are automatically calculated and applied based on host memory capabilities both at first installation and update time

* Installer: improved the Grid Managers selection page to let users seamlessly configure HPC scheduler and VDI Session Managers
* Installer: improved behavior during EF Portal updates
* Logs: ef.log, agent.remote.log and dcvsm.log do not show error stack traces for better readability. Logs with full stack traces are now collected into corresponding .withtrace.log files

* Logs: Tomcat logging strategy has been revised to control growth of the Tomcat logs directory
* Logs: cleaned up misleading Hazelcast warnings

BUGFIXES:

* SLURM plugin: fixed a bug when the cluster name contained hyphens
* Login: fixed 'Access not authorized' bug happening sometimes after session expiration or EF Portal restart
* UI: fixed bar charts display
* Installer: fixed Nashorn deprecation warning

Date: September 20, 2024 - Version 2024.0

ENHANCEMENTS:

* Added support for Redhat 8 and 9, CentOS 8 and 9, Rocky 8 and 9, AlmaLinux 8 and 9
* Added support Ubuntu 22, 24
* Added support Amazon Linux 2023
* Added support for SLURM 22.x, 23.x
* Support for file, ssh and s3 remote spoolers
* Support for Copy/Paste/Move operations on files and directories between spoolers, remote spoolers and places
* Display of present copy/paste/move operation
* Preview of data in spoolers supported for images, text and pdf as overlay in the file manager, draggable and resizable
* Remember last location in a file manager and spooler and return to that path when viewing the “place”
* Show history of paths in places (spooler, filemanager) and allow user to jump to a selected previous locations
* Support for UTF-8 Characters in Filenames
* Custom Places: Option to include further quick-access places via ~/.ef/ef.places
* Job History Service displaying history of job usage in table and graph view
* DCV SM Integration - Added Enqueue API support with default to enqueue DCV SM sessions
* DCV SM Integration - Added option for command line parameters for the autorun delegate script
* Show full hostname in title of hostview when hovering over hostname

CHANGES:

* Updated Apache Commons Text from 1.9 to 1.11
* Upgrade jQuery UI to v1.13.2
* Upgrade jqGrid to v4.7

Date: August 3, 2023 - Version 2021.0-r1667

CHANGES:

* EnginFrame now ships with Tomcat® 9.0.78 (from 9.0.64)
* EnginFrame now ships with Apache® Commons Text 1.10.0 (from 1.9).
* EnginFrame now ships with Gson 2.8.9 (from 2.8.x).
* EnginFrame documentation added: Restarting DCV Session Manager

Date: August 9, 2022 - Version 2021.0-r1657

CHANGES:
* If updating from a previous EnginFrame instance, you must change the server.xml file as described in Updating to EnginFrame version 2021.0-r1653 or later.
* Fixed a bug to prevent Tomcat use of HTTPS.

BUGFIXES:
* Fixed a bug to prevent Tomcat use of HTTPS.
* Fixed a bug that occurred with the use of AWS Cognito as the authentication server for DCVSM.

Date: August 5, 2022 - Version 2021.0-r1653

ENHANCEMENTS:
* EnginFrame now ships with Tomcat® 9.0.64.Bug fixes:

BUGFIXES:
* Fixed a bug that caused global actions not to be shown in spooler view in some cases.