EVIDEN

Identity and Access Management

DirX Identity

Monitoring DirX Identity Servers with Nagios

Version 9.0.2, Edition November 2026



All product names quoted are trademarks or registered trademarks of the manufacturers concerned.

© 2026 Atos Group

All Rights Reserved

Distribution and reproduction not permitted without the consent of Atos Group.

Table of Contents

Copyright ii
Preface 1
DirX Identity Documentation Set 2
Notation Conventions 4
1. Overview 6
1.1. Background Documentation 6
1.2. DirX Identity Plugins 7
1.3. Configuration Scenarios 7
1.4. What to Check in the DirX Identity Server Environment 9
2. Configuration 10
2.1. JAVA 11 Support 10
2.2. Deploying the DirX Identity Plugins 11
2.3. Configuring the JNRPE Server 11
2.3.1. CHECK_ADAPTOR and CHECK_ADAPTORBYNAME Commands 12
2.3.1.1. Command Definitions 12
2.3.1.2. Arguments 12
2.3.1.2.1. Examples 12
2.3.2. CHECK_CSERV and CHECK_CSERVATT Commands 14
2.3.2.1. Command Definitions 14
2.3.2.2. Arguments 14
2.3.2.3. Examples 14
2.3.3. CHECK_IDSJ Command 15
2.3.3.1. Command Definition 15
2.3.3.2. Arguments 15
2.3.3.3. Examples 15
2.3.4. CHECK_JMXMEM Command 16
2.3.4.1. Command Definition 16
2.3.4.2. Arguments 16
2.3.5. CHECK_TOMCAT Command 16
2.3.6. CHECK_WF and CHECK_WF_EXC Commands 16
2.3.6.1. Command Definitions 16
2.3.6.2. Arguments 17
2.3.6.3. Examples 17
2.3.7. Identifying Java-based Server Attributes to Check 18
2.3.8. Identifying the JMX Ports of DirX Identity Servers 19
2.4. Configuring Nagios 20
2.4.1. Defining the JNRPE Call 20
2.4.2. Defining the Services 20
Legal Remarks 23

Preface

This document presents a use case that illustrates how to set up Nagios to monitor DirX Identity Servers.

- Chapter 1 provides general information about the use case.
- Chapter 2 describes how to set up the JNRPE and Nagios configuration.

DirX Identity Documentation Set

Version: 9.0.2 | Date: 2026-11-02 | Timestamp: 13:57:36 UTC

The DirX Identity document set consists of the following manuals:

- *DirX Identity Introduction*. Use this book to obtain a description of DirX Identity architecture and components.
- *DirX Identity Release Notes*. Use this book to understand the features and limitations of the current release. This document is shipped with the DirX Identity installation as the file **release-notes.pdf**.
- *DirX Identity History of Changes*. Use this book to understand the features of previous releases. This document is shipped with the DirX Identity installation as the file **history-of-changes.pdf**.
- *DirX Identity Tutorial*. Use this book to get familiar quickly with your DirX Identity installation.
- *DirX Identity Provisioning Administration Guide*. Use this book to obtain a description of DirX Identity provisioning architecture and components and to understand the basic tasks of DirX Identity provisioning administration using DirX Identity Manager.
- *DirX Identity Connectivity Administration Guide*. Use this book to obtain a description of DirX Identity connectivity architecture and components and to understand the basic tasks of DirX Identity connectivity administration using DirX Identity Manager.
- *DirX Identity User Interfaces Guide*. Use this book to obtain a description of the user interfaces provided with DirX Identity.
- *DirX Identity Application Development Guide*. Use this book to obtain information how to extend DirX Identity and to use the default applications.
- *DirX Identity Customization Guide*. Use this book to customize your DirX Identity environment.
- *DirX Identity Integration Framework*. Use this book to understand the DirX Identity framework and to obtain a description how to extend DirX Identity.
- *DirX Identity Web Center Reference*. Use this book to obtain reference information about the DirX Identity Web Center.
- *DirX Identity Web Center Customization Guide*. Use this book to obtain information how to customize the DirX Identity Web Center.
- *DirX Identity Resolution Service Setup Guide*. Use this book to set up the DirX Identity Resolution Service as a standalone service.
- *DirX Identity Meta Controller Reference*. Use this book to obtain reference information about the DirX Identity meta controller and its associated command-line programs and files.
- *DirX Identity Connectivity Reference*. Use this book to obtain reference information about the DirX Identity agent programs, scripts, and files.
- *DirX Identity Troubleshooting Guide*. Use this book to track down and solve problems in your DirX Identity installation.

- *DirX Identity Installation Guide*. Use this book to install DirX Identity.
- *DirX Identity Migration Guide*. Use this book to migrate from previous versions.
- *DirX Identity REST Service Guide*. Use this book to migrate from previous versions.

Notation Conventions

Boldface type
In command syntax, bold words and characters represent commands or keywords that must be entered exactly as shown.

In examples, bold words and characters represent user input.

Italic type
In command syntax, italic words and characters represent placeholders for information that you must supply.

[ ]
In command syntax, square braces enclose optional items.

{ }
In command syntax, braces enclose a list from which you must choose one item.

In Tcl syntax, you must actually type in the braces, which will appear in boldface type.

|
In command syntax, the vertical bar separates items in a list of choices.

...
In command syntax, ellipses indicate that the previous item can be repeated.

userID_home_directory
The exact name of the home directory. The default home directory is the home directory of the specified UNIX user, who is logged in on UNIX systems. In this manual, the home pathname is represented by the notation *userID_home_directory*.

install_path
The exact name of the root of the directory where DirX Identity programs and files are installed. The default installation directory is *userID_home_directory*/**DirX Identity** on UNIX systems and **C:\Program Files\DirX\Identity** on Windows systems. During installation the installation directory can be specified. In this manual, the installation-specific portion of pathnames is represented by the notation *install_path*.

dirx_install_path
The exact name of the root of the directory where DirX programs and files are installed. The default installation directory is *userID_home_directory*/**DirX** on UNIX systems and **C:\Program Files\DirX** on Windows systems. During installation the installation directory can be specified. In this manual, the installation-specific portion of pathname is represented by the notation *dirx_install_path*.

dxi_java_home
The exact name of the root directory of the Java environment for DirX Identity. This location is specified while installing the product. For details see the sections "Installation" and "The Java for DirX Identity".

tmp_path

The exact name of the tmp directory. The default tmp directory is /tmp on UNIX systems. In this manual, the tmp pathname is represented by the notation *tmp_path*.

tomcat_install_path
The exact name of the root of the directory where Apache Tomcat programs and files are installed. This location is defined during product installation.

mount_point
The mount point for DVD device (for example, **/cdrom/cdrom0**).

1. Overview

The DirX Identity servers provide Java interfaces that permit them to be monitored via Java Management Extensions (JMX) technology. So do other Java applications that run in the DirX Identity environment, like the Apache Tomcat servlet container and the Apache ActiveMQ™ message broker. Because Nagios® IT monitoring software - a popular tool for monitoring the status of hardware, applications and services - supports JMX monitoring, it can be set up to monitor the state of DirX Identity servers and other Java applications and alert administrators when problems appear and also when problems are resolved.

Nagios supports the concept of "plugins" for managing the details of monitoring the different resources available in a given configuration: when provided with a set of input parameters, a plugin gathers some metric, compares it with the passed-in thresholds, and returns a status code (OK, WARNING, CRITICAL, or UNKNOWN) and an explanatory message. DirX Identity provides a set of Java-based plugins for checking the status of DirX Identity servers that can be integrated into the Nagios environment.

However, because Nagios calls to Java-based plugins can generate a significant amount of resource overhead (the Nagios Remote Procedure Executor (NRPE) starts a Java Virtual Machine (JVM) for every Java-based plugin execution), efficient use of Nagios for DirX Identity monitoring means integrating the Java NRPE (JNRPE) add-on into the Nagios environment. JNRPE is a replacement for NRPE that allows for execution of Nagios plugins on remote machines without incurring the overhead of many short-lived JVM instances.

The rest of this chapter describes the DirX Identity plugins for checking DirX Identity server status and the supported use case configuration scenarios. See the section "Background Documentation" for references to documents and links that provide more detail about Nagios and JNRPE functions, installation, configuration and plugins.

1.1. Background Documentation

The following documents and links provide additional details about the concepts and procedures referenced in this use case document. We recommend that you become familiar with the information in these documents before proceeding with the tasks described in this use case document.

- **http://www.nagios.org/** - provides information about Nagios software installation and configuration and Nagios plugins
- **http://www.jnrpe.it/** - provides information about JNRPE software installation and configuration and how to create JNRPE plugins.
- **http://jnrpe.sourceforge.net/jnrpe-plugins** - provides information about for JNRPE plugins, such as CHECK_JMX and CHECK_TOMCAT
- **https://sourceforge.net/projects/yajsw/** - provides information about the Java Service Wrapper YAJSW
- **http://docs.oracle.com/javase/8/docs/technotes/guides/management/jconsole.html** - provides usage information about Oracle's JMX console (Jconsole)
- *DirX Identity Connectivity Administration Guide* → chapter title "Managing DirX Identity

Servers" - provides detailed information about managing the Java-based and C++-based Servers.

You can also consult the online help provided with DirX Identity Manager (click **Help** in a dialog or topic to access the online help).

1.2. DirX Identity Plugins

DirX Identity provides the following DirX Identity-specific plugins to be used by the JNRPE Server:

- DXI_CHECK_JMX - a plugin for accessing the Java-based Identity Server and other JMX-enabled programs. This plugin is an extension of the standard JNRPE plugin CHECK_JMX that supports:
  - Extended values for warning and critical thresholds. It supports the threshold syntax described in http://nagios-plugins.org/doc/guidelines.html#THRESHOLDFORMAT You can define the critical range to be less than or equal to 3 (:3) and the warning range to be less than or equal to 6 (:6), because 0 is the worst state and 10 is the best state for the Java-based Server. The standard CHECK_JMX plugin does not support this syntax.
  - Severity levels for exceptions, which define whether an exception (for example, "counter not available" or "server not reachable") should be treated as UNKNOWN (**u**), CRITICAL (**c**), WARNING (**w**) or OK (**o**). For example, suppose you want to check a workflow counter like "number of erroneous workflow runs". If the workflow has never been started, no counter is available and an exception is thrown; it's also thrown if the server is down or unreachable. In these instances, you don't want to see an error, because nothing has gone wrong. To avoid this problem, you can use the severity level argument to specify that these cases should be treated as "OK". See the CHECK_WF_EXC command description in the section "CHECK_WF and CHECK_WF_EXC Commands" for more detail.
- DXI_CHECK_CSERVER - a plugin for accessing the C++-based Identity Server using internal DirX Identity interfaces.

1.3. Configuration Scenarios

The services to be deployed to support this use case are:

- Nagios Core™, version 4.0.8
- JNRPE Server, version 2.0.5
- JNRPE plugins, version 2.0.5
- YAJSW 12.16
- DirX Identity Servers, V8.10

You can run Nagios and the JNRPE Server on the same host or on different hosts. If the JNRPE Server is located on another host, the JNRPE port (usually **5666**) must be accessible.

Nagios can access the JNRPE Server over a plain or an SSL-enabled connection via a call to **jcheck_nrpe/s**, which is a Java implementation of the Nagios **check_nrpe** utility. The next two figures illustrate these configurations.

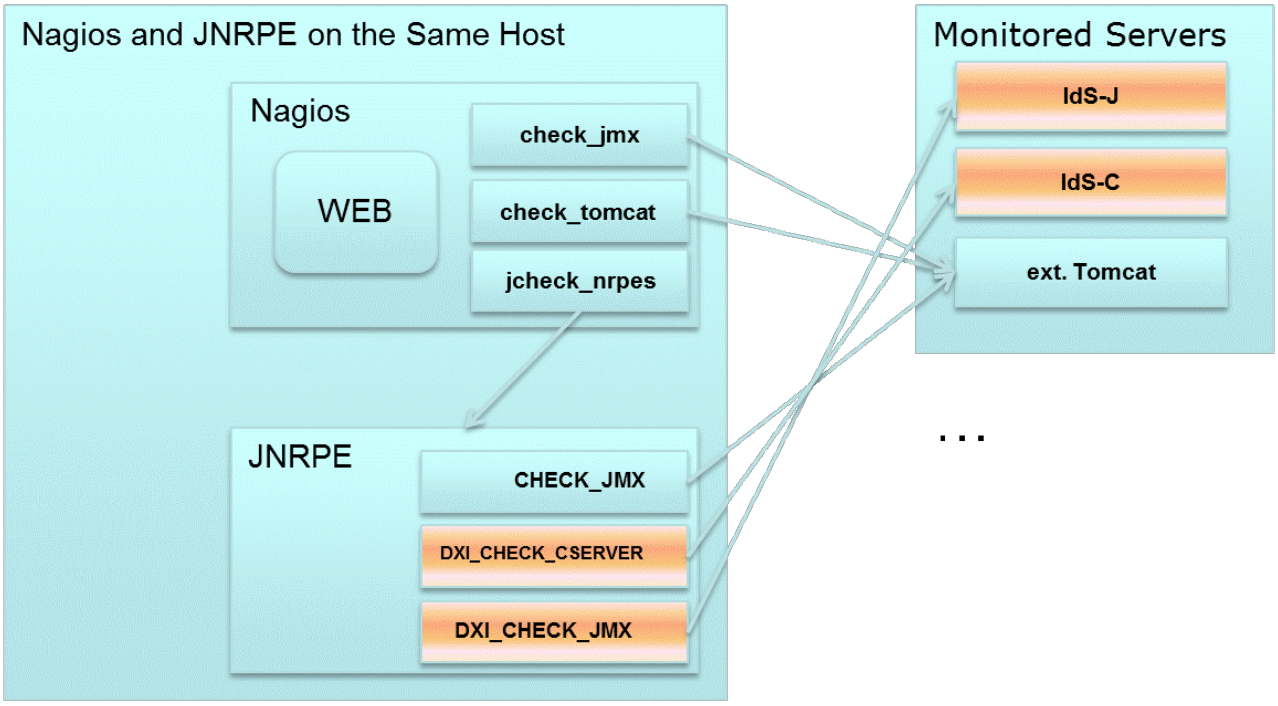


Figure 1. Nagios and JNRPE on the Same Host with SSL Connection

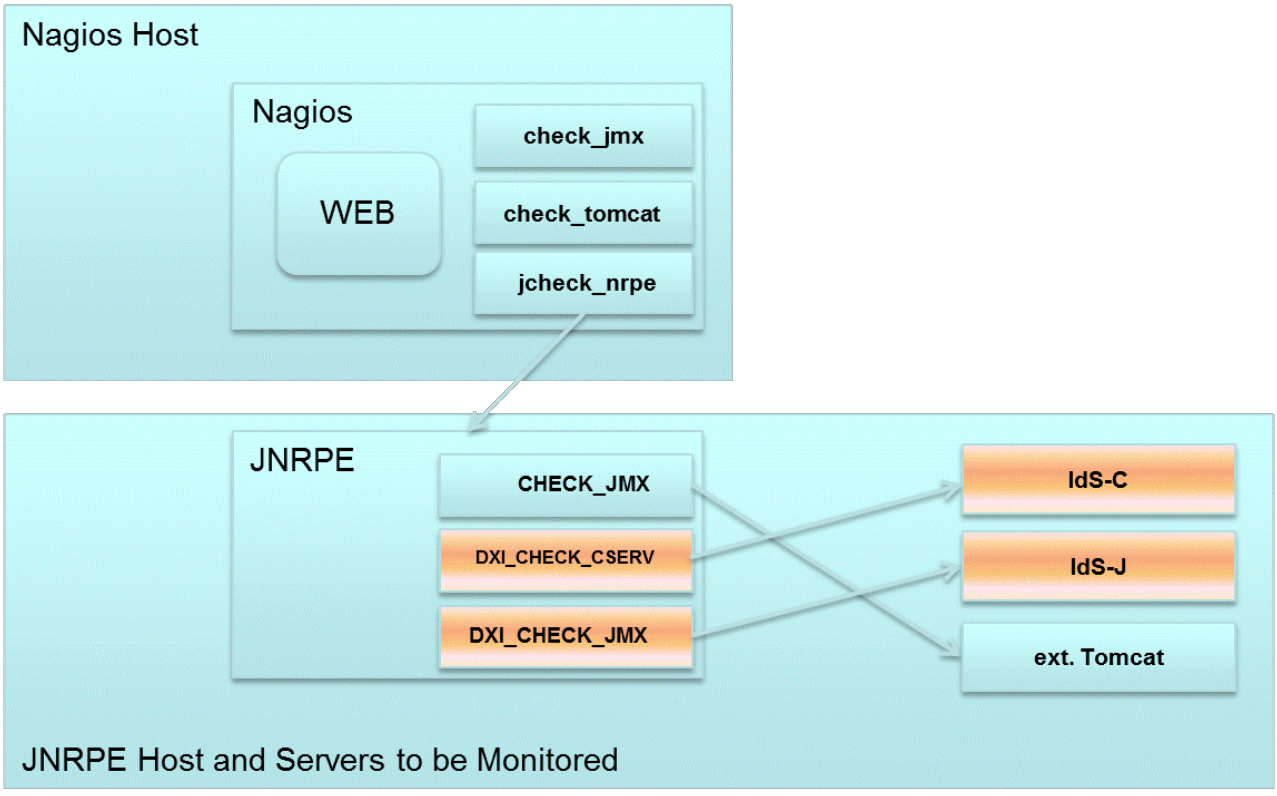


Figure 2. Nagios and JNRPE on Different Hosts with Plain Connection

See the **jcheck_nrpe** documentation for details about its command line format. The section "Defining the JNRPE Call" in this use case provides information on how to configure the call

in the Nagios environment.

You can use the JNRPE Server to monitor DirX Identity servers running on the same host or on remote hosts. If you want to monitor servers on other hosts, access to the port for JMX access on these hosts must be allowed.

1.4. What to Check in the DirX Identity Server Environment

The most important items to check are:

- The state of a Java-based Server. See the section "CHECK_IDSJ Command" for details.
- The state of a C++-based Server. See the CHECK_CSERV command in the section "CHECK_CSERV and CHECK_CSERVATT Commands" for details.
- The outstanding messages of a specific adaptor running in a Java-based Server. See the CHECK_ADAPTORBYNAME command in the section "CHECK_ADAPTOR and CHECK_ADAPTORBYNAME Commands" for details.

For the Java-based Server, many additional checks can be configured. The CHECK_WF_EXC command described in "CHECK_WF and CHECK_WF_EXC Commands" is one example. The section “Identifying Java-based Server Attributes to Check" describes how to determine other Java-based server attributes to check. You can also use all of the standard checks provided with Nagios and JNRPE, such as CHECK_JMXMEM.

Here is an example of a status report displayed in the Nagios web interface from a check of a C++-based Server and a Java-based Server:

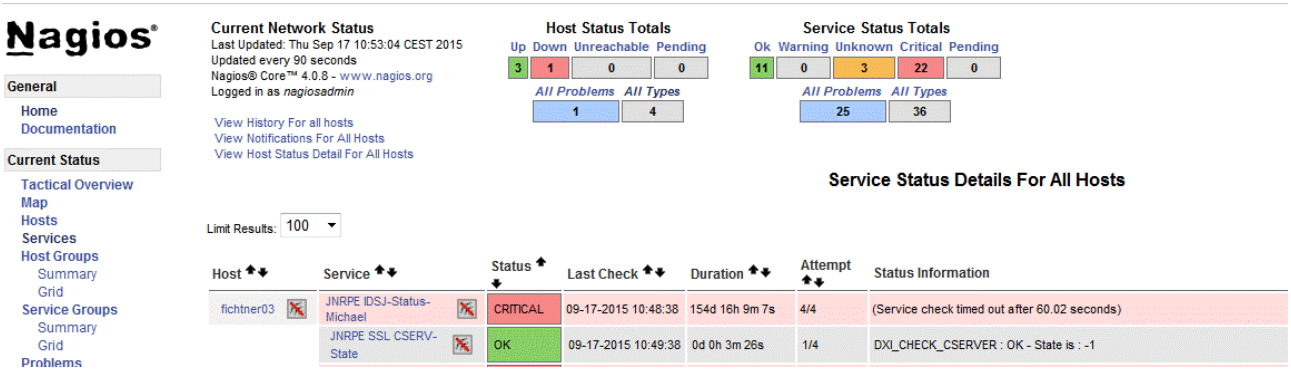


Figure 3. Example Nagios Status Report on DirX Identity Servers

In the example display, the C++-based Server is running, but the Java-based Server is unreachable.

2. Configuration

To configure your Nagios and JNRPE environments to monitor DirX Identity servers:

- Deploy the DirX Identity plugins to the JNRPE server.
- Configure the JNRPE and Nagios servers to support DirX Identity plugin execution.

2.1. JAVA 11 Support

Currently, no JNRPE for JAVA11 is available.

Starting the JNRPE server gives some warnings with JAVA11:



An illegal reflective access operation has occurred



Illegal reflective access by io.netty.util.internal.PlatformDependent0(file:/D:/Program%20Files/JNRPE/lib/netty-all-4.0.19.Final.jar) to field java.nio.Buffer.address



Please consider reporting this to the maintainers of io.netty.util.internal.PlatformDependent0



Use --illegal-access=warn to enable warnings of further illegal reflective access operations



All illegal access operations will be denied in a future release

You can suppress these warnings by using the following options in the java call:

--add-opens java.base/java.nio=ALL-UNNAMED --add-opens java.base/sun.nio.ch=ALL-UNNAMED

For example, add these options to the %JAVACMD% line in the jnrpe.bat file.

Starting could also fail, because of the included YAJSW 11.04. You must replace the version 11.04 with version 12.16:

1. Download YAJSW 12.6 from here: https://sourceforge.net/projects/yajsw/files/yajsw/yajsw-stable-12.16/yajsw-stable-12.16.zip/download
2. Rename the folder *JNRPE_install_path*/**wrapper** to *JNRPE_install_path*/**wrapper-11.04**
3. Unzip the file yajsw-stable-12.16.zip.
4. Rename the new folder **yajsw-stable-12.16** to **wrapper**.
5. Move this folder **wrapper** to *JNRPE_install_path*/
6. Copy the folder *JNRPE_install_path*/**wrapper-11.04/etc** into the new folder

JNRPE_install_path/**wrapper/**.

If a JNRPE version supporting JAVA11 is available these workarounds are obsolete.

2.2. Deploying the DirX Identity Plugins

The DirX Identity installation provides the JNRPE plugin folder in *dxi_install_path***/nagiosSupport/dxiplugin**. All of the required jar files are included in this folder.

To deploy the plugins to your JNRPE environment, simply copy this folder to the *JNRPE_install_path*/**plugins** subdirectory. Refer to the JNRPE documentation for details about JNRPE server directory structure.

Deploying the JNRPE plugins on Linux works without any further action.

On Windows, you may need to specify the CLASSPATH_PREFIX environment variable. Refer to all jar files in the *JNRPE_install_path*/**plugins** subdirectory.

2.3. Configuring the JNRPE Server

In Nagios and JNRPE, a "command" is an instance of a plugin that is configured with an explicit command line to invoke the plugin to perform a specific task. The JNRPE configuration file *JNRPE_install_path***/etc/jnrpe.ini** is the repository for all of the commands available for execution by the JNRPE Server.

Each command definition follows the format:

Command Name : *Plugin Name* : *Command Line*

In *Command Line* parameters, you can supply hard-coded values or variables in the form **$ARG*n***$* to be resolved with specific values at execution time.

See the JNRPE documentation for complete details about JNRPE command syntax.

To configure your JNRPE installation to monitor DirX Identity servers with the DirX Identity Nagios plugins, update the *JNRPE_install_path***/etc/jnrpe.ini** file in your JNRPE server environment to define your JNRPE commands. These commands can call the DirX Identity plugins or standard JNRPE plugins to check DirX Identity servers and other JMX-enabled servers like Tomcat and ActiveMQ.

The connection to the JMX interfaces are done with SSL/TLS. Make sure that the necessary certificate of the certification authority, which has signed the server certificates of the JMX interfaces, are included in the cacerts certificate store of the java implementation, which is used for running the JRNPE server.

We provide a sample **jnrpe.ini** file in *dxi_install_path*/**nagiosSupport**]**/samples/JNRPE**.

The next sections describe the sample command definitions given in this file and show examples of their use when run from the JNRPE interactive console.

2.3.1. CHECK_ADAPTOR and CHECK_ADAPTORBYNAME Commands

These commands check the outstanding responses of a specific JMS adaptor. The sample CHECK_ADAPTOR command is configured to check the ProvisioningRequestListener (see the **--object** parameter). The CHECKADAPTORBYNAME command accepts a JMS adaptor name as an argument.

2.3.1.1. Command Definitions

```
CHECK_ADAPTOR : DXI_CHECK_JMX --url
service:jmx:rmi:///jndi/rmi://$ARG1$:$ARG2$/jmxrmi --object
com.siemens.idm:type=idsj,topic=adaptor,name=ProvisioningRequestListe
ner --attribute outstandingResponses --warning $ARG3$ --critical
$ARG4$ --username domainadmin --password $ARG5$

CHECK_ADAPTORBYNAME : DXI_CHECK_JMX --url
service:jmx:rmi:///jndi/rmi://$ARG1$:$ARG2$/jmxrmi --object
com.siemens.idm:type=idsj,topic=adaptor,name=$ARG3$ --attribute
outstandingResponses --warning $ARG4$ --critical $ARG5$ --username
domainadmin --password $ARG6$
```

2.3.1.2. Arguments

The CHECK_ADAPTOR command takes the following arguments:

- ARG1 - Host.
- ARG2 - JMX port.
- ARG3 - Warning level.
- ARG4 - Critical level.
- ARG5 – Password for JMX authentication with user domainadmin.

The CHECKADAPTORBYNAME command takes the following arguments:

- ARG1 - Host.
- ARG2 - JMX port.
- ARG3 – Name of the JMS adaptor to check.
- ARG4 - Warning level.
- ARG5 - Critical level.
- ARG6 – Password for JMX authentication with user domainadmin.

2.3.1.2.1. Examples

This example shows that there are no outstanding messages for the

ProvisioningRequestListener adaptor by using the low water mark (900) as the warning level and the high water mark (1000) as the critical level:

```
JNRPE> command:CHECK_ADAPTOR localhost 39999 900: 1000: dirx
Value: 0
DXI_CHECK_JMX : OK - outstandingResponses is :
0|State=0,000000;900:;1000:;0,00000
0
```

This example simulates a warning by defining the warning level to 0:

```
JNRPE> command:CHECK_ADAPTOR localhost 39999 0: 1000: dirx
Value: 0
DXI_CHECK_JMX : WARNING - outstandingResponses is :
0|State=0,000000;0:;1000:;0,00
0000:
```

This example uses CHECK_ADAPTORBYNAME to check the EntryChangeListener adaptor:

```
JNRPE> command:CHECK_ADAPTORBYNAME localhost 39999
EntryChangeListener 900: 1000 dirx
:
Value: 0
DXI_CHECK_JMX : OK - outstandingResponses is :
0|State=0,000000;900:;1000:;0,00000
0
```

This example generates an error for CHECK_ADAPTORBYNAME by using the unknown adapter name **EntryChangeListenerxx**:

```
JNRPE> command:CHECK_ADAPTORBYNAME localhost 39999
EntryChangeListenerxx 900: 1000: dirx
DXI_CHECK_JMX : CRITICAL - An error has occurred during execution of
the DXI_CHECK_JMX plugin returning 0 , error :
com.siemens.idm:type=idsj,topic=adaptor,name=EntryChangeListenerxx
JNRPE>
```

2.3.2. CHECK_CSERV and CHECK_CSERVATT Commands

These commands check the state of the C++-based Server (IdS-C) (CHECK_CSERV) or the specified C++-based Server attribute (CHECK_CSERVATT). The State attribute is currently the only C++-based Server attribute that is meaningful to check.

2.3.2.1. Command Definitions

```
CHECK_CSERV : DXI_CHECK_CSERVER --host $ARG1$ --port $ARG2$
--attribute State --warning $ARG3$ --critical $ARG4$

CHECK_CSERVATT : DXI_CHECK_CSERVER --host $ARG1$ --port $ARG2$
--attribute $ARG3$ --warning $ARG4$ --critical $ARG5$
```

2.3.2.2. Arguments

The CHECK_CSERV command takes the following arguments:

- ARG1 - Host.
- ARG2 - JMX port. See the section "Identifying the JMX Ports of DirX Identity Servers" for information on how to determine a C++-based Server's JMX port.
- ARG3 - Warning level.
- ARG4 - Critical level.

2.3.2.3. Examples

Here is an example of the CHECK_CSERV command that checks whether or not the server is available:

```
JNRPE> command:CHECK_CSERV  localhost  5315 :6 ~:3
Value: 10
DXI_CHECK_CSERVER : OK - State is : 10|State=10,000000;:6;~:3;-
2,000000
```

Here is an example of the CHECK_CSERVATT that provides State as the attribute name:

```
JNRPE> command:CHECK_CSERVATT  localhost  5315 State :6 ~:3
Value: 10
DXI_CHECK_CSERVER : OK - State is : 10|State=10,000000;:6;~:3;-
2,000000
```

2.3.3. CHECK_IDSJ Command

This command checks the state of the Java-based Server (IdS-J).

2.3.3.1. Command Definition

```
CHECK_IDSJ : DXI_CHECK_JMX --url
service:jmx:rmi:///jndi/rmi://$ARG1$:$ARG2$/jmxrmi --object
com.siemens.idm:type=idsj,topic=core,name=Server --attribute State
--warning $ARG3$ --critical $ARG4$ --username domainadmin --password
$ARG5$
```

2.3.3.2. Arguments

This command takes the following arguments:

- ARG1 - Host.
- ARG2 - JMX port.
- ARG3 - Warning level.
- ARG4 - Critical level.
- ARG5 – Password for JMX authentication with user domainadmin.

2.3.3.3. Examples

Here is an example command that shows that the state of the Java-based Server is good:

```
JNRPE> command:CHECK_IDSJ localhost 39999 :6  :3 dirx
Value: 8
DXI_CHECK_JMX : OK - State is : 8|State=8,000000;:6;:3;0,000000
```

Here is an example command that shows that Java-based Server is down:

```
JNRPE> command:CHECK_IDSJ localhost 39999 :6  :3 dirx
DXI_CHECK_JMX : CRITICAL - An error has occurred during execution of
the DXI_CHECK_JMX plugin returning 0 , error : Failed to retrieve
RMIServer stub: javax.naming.ServiceUnavailableException [Root
exception is java.rmi.ConnectException: Connection refused to host:
localhost; nested exception is:
    java.net.ConnectException: Connection refused: connect]
JNRPE>
```

2.3.4. CHECK_JMXMEM Command

This command uses the standard JNRPE plugin CHECK_JMX to check the standard JVM memory usage attribute. You can use this command to check any Java application; you only need to select the correct JMX port. The section "Identifying the JMX Ports of DirX Identity Servers" provides instructions for determining the JMX ports for DirX Identity Servers.

2.3.4.1. Command Definition

```
CHECK_JMXMEM : CHECK_JMX --url
service:jmx:rmi:///jndi/rmi://$ARG1$:$ARG2$/jmxrmi --object
java.lang:type=Memory --attribute HeapMemoryUsage -K used -I
HeapMemoryUsage -J used --warning $ARG3$ --critical $ARG4$ --username
domainadmin --password $ARG5$
```

2.3.4.2. Arguments

The CHECK_JMXMEM command takes the following arguments:

- ARG1 - Host.
- ARG2 - JMX port.
- ARG3 - Warning level.
- ARG4 - Critical level.
- ARG5 – Password for JMX authentication with user domainadmin (optional).

2.3.5. CHECK_TOMCAT Command

This command uses the standard JNRPE plugin CHECK_TOMCAT to check the Tomcat server. See the CHECK_TOMCAT JNRPE plugin documentation for details.

2.3.6. CHECK_WF and CHECK_WF_EXC Commands

These sample commands check a statistics attribute of the specified workflow. The command definitions given here specify the **Export Users to LDIF** workflow in the **--object** parameter. When specifying a different workflow name, be sure to enclose it in double quotation marks (") because workflow names contain blank spaces.

2.3.6.1. Command Definitions

```
CHECK_WF : DXI_CHECK_JMX --url
service:jmx:rmi:///jndi/rmi://$ARG1$:$ARG2$/jmxrmi --object
"com.siemens.idm:type=idsj,topic=core,name=Statistics,name0=My-
Company,name1=Provisioning,name2=Source
```

```
Realtime,name3=LDIFfile,name4=Export Users to LDIF" --attribute
$ARG3$ --warning $ARG4$ --critical $ARG5$ --username domainadmin
--password $ARG6$

CHECK_WF_EXC : DXI_CHECK_JMX --url
service:jmx:rmi:///jndi/rmi://$ARG1$:$ARG2$/jmxrmi --object
"com.siemens.idm:type=idsj,topic=core,name=Statistics,name0=My-
Company,name1=Provisioning,name2=Source
Realtime,name3=LDIFfile,name4=Export Users to LDIF" --attribute
$ARG3$ --warning $ARG4$ --critical $ARG5$ --ExceptionState $ARG6$
--username domainadmin --password $ARG7$
```

2.3.6.2. Arguments

The CHECK_WF and CHECK_WF_EXC commands take the following arguments:

- ARG1 - Host.
- ARG2 - JMX port.
- ARG3 - Attribute name (for example, join.SUCCEEDED, join.AddFailed).
- ARG4 - Warning level.
- ARG5 - Critical level.
- ARG6 (CHECK_WF_EXC only) - Exception definition. Defines whether an exception (for example, "counter not available" or "server not reachable") should be treated as UNKNOWN (**u**), CRITICAL (**c**), WARNING (**w**) or OK (**o**).
- ARG6 (CHECK_WF only) – Password for JMX authentication with user domainadmin.
- ARG7 (CHECK_WF_EXC only) – Password for JMX authentication with user domainadmin.

2.3.6.3. Examples

This example checks the join.SUCCEEDED attribute and sets the warning and critical values to simulate a warning:

```
JNRPE>  command:CHECK_WF localhost 39999 join.SUCCEEDED 1: 2: dirx
Value: 1
DXI_CHECK_JMX : WARNING - join.SUCCEEDED is :
1|State=1,000000;1:;2:;0,000000
JNRPE>
```

This example checks the join.SUCCEEDED attribute and shows that the workflow has never been started because the counter is not available. Because **u** is the specified exception definition, UNKNOWN is returned:

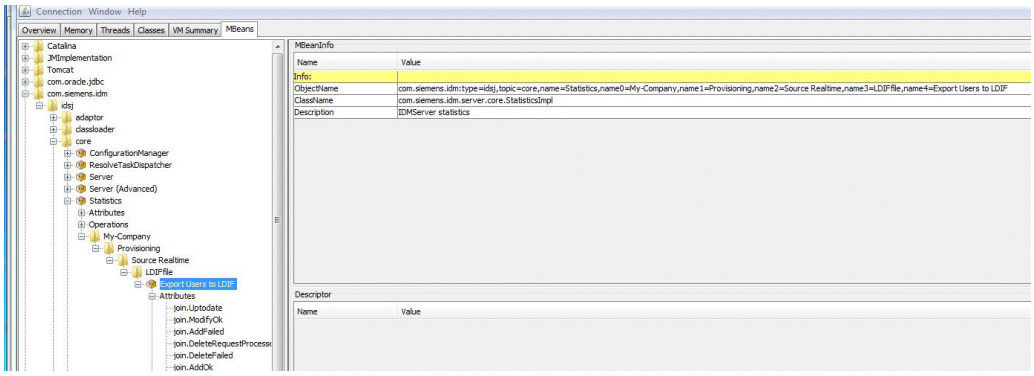
```
JNRPE>  command:CHECK_WF_EXC localhost 39999 join.SUCCEEDED 1: 2: u
dirx
DXI_CHECK_JMX : UNKNOWN - An error has occurred during execution of
the DXI_CHEC
K_JMX plugin returning 0 , error :
com.siemens.idm:type=idsj,topic=core,name=Sta
tistics,name0=My-Company,name1=Provisioning,name2=Source
Realtime,name3=LDIFfile
,name4=Export Users to LDIF
```

2.3.7. Identifying Java-based Server Attributes to Check

You can use the standard JMX console (JConsole) graphical monitoring tool included in the Java Development Kit (JDK) to identify the name of the Java-based Server attribute you want to check. For example, let's look at the CHECK_WF command. The workflow object name configured in this command is:

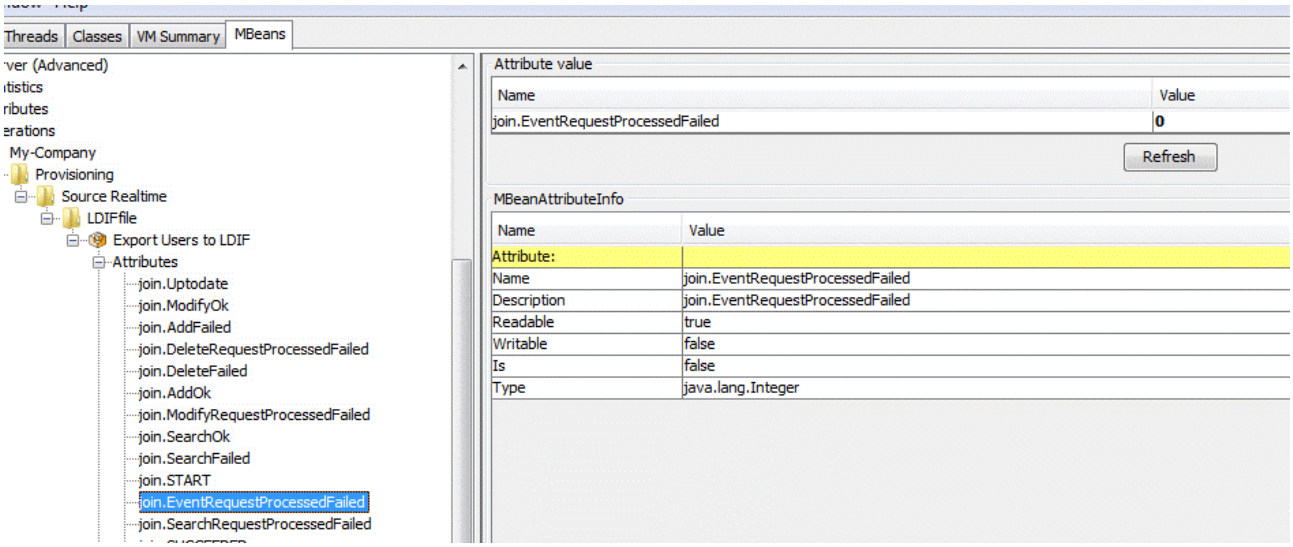
```
jmxrmi --object
"com.siemens.idm:type=idsj,topic=core,name=Statistics,name0=My-
Company,name1=Provisioning,name2=Source
Realtime,name3=LDIFfile,name4=Export Users to LDIF"
```

This workflow name is displayed in JConsole at the following location:



In the **Name** column of the right-hand pane, **ObjectName** contains the name needed for configuring the DXI_CHECK_JMX plugin's jmxrmi **--object** parameter.

The attribute name (here, join.EventRequestProcessedFailed) can be found in JConsole at this location:



In the Attribute value section, the **Name** column contains the attribute name needed for configuration.

2.3.8. Identifying the JMX Ports of DirX Identity Servers

To obtain the JMX port for a Java-based Server, examine the following parameter in the INI file *dxi_install_path*/**ids-j-***domain*-S*n*/**bin/idmsvc.ini**:

-DIDM_JMXPORT=40005

For example:

```
15=-DIDM_JMXPORT=40005
```

For more information, see the section "Java-based Server INI File Parameters" in the chapter "Managing DirX Identity Servers" in the *DirX Identity Connectivity Administration Guide*.

Or look in the configuration.ini file in *dxi_install_path* for parameter IdS-J.jmx_port=40005

To obtain the JMX port for a C++-based Server, use the DirX Identity Manager:

- Log in to the DirX Identity Manager and select the Connectivity View.
- In the Connectivity View, select the C Server object in the tree (in DirX Identity Servers → C Servers) and then click the JMX Access tab.
- In **JMX Service**, follow the link to the JMX Service object (by clicking the Properties icon) and then use the value given in **Data Port** (in the **Connection Parameters** section).

For more information, use the DirX Identity Manager context-sensitive help.

2.4. Configuring Nagios

To configure a Nagios installation for JNRPE plugin execution:

- Define the call to the JNRPE server in *nagios_install_path***/nagios/etc/objects/commands.cfg**
- Define the services you want to make available from the Nagios Web interface in *nagios_install_path***/nagios/etc/objects/localhost.cfg**

We provide some Nagios configuration files in *dxi_install_path***/nagiosSupport/samples/**. The **JNRPE** subfolder contains JNRPE configuration samples, while the **Nagios** subfolder contains samples for the Nagios configuration.

2.4.1. Defining the JNRPE Call

In the Nagios **commands.cfg** file, define the call to the JNRPE server. Provide the command name and the command line. Here is an example:

```
define command{
        command_name    jcheck_nrpes
        command_line    $USER1$/jcheck_nrpe -H $ARG1$ -p 5667 -t
$ARG2$ -c $ARG3$ -a $HOSTADDRESS$ $ARG4$ $ARG5$ $ARG6$ $ARG7$ $ARG8$
$ARG9$
        }
```

See the Nagios documentation for a description of command definition format.

2.4.2. Defining the Services

In the Nagios **localhost.cfg** file, define the JNRPE commands that you want to make available as "services" from the Nagios Web interface. Provide a name for the service, specify the JNRPE command name you defined in the JNRPE server **jnrpe.ini** file that Nagios is to run and provide values for arguments to the command line.

This example defines a service named "JNRPE SSL IDSJ-Status" that calls the CHECK_IDSJ JNRPE command to check the state of an IdS-J server running on a host (the given password is used to authenticate via JMX):

```
define service{
        use                             local-service     ; Name of
service template to use
        host_name                       hostname
        service_description             JNRPE SSL IDSJ-Status
        check_command
jcheck_nrpes!JNRPE_host!10!CHECK_IDSJ!40005!:6!:3!<password for JMX
```

```
authentication to ids-j>
        notifications_enabled           0
        }
```

This example defines a service named "JNRPE SSL MEM on IDSJ" that calls the CHECK_JMXMEM JNRPE command to check the standard JVM memory usage attribute for java-based DirX Identity server running on a remote host:

```
define service{
        use                             local-service         ; Name
of service template to use
        host_name                       distributed_host_A
        service_description             JNRPE SSL MEM on IDSJ
        check_command
jcheck_nrpes!JNRPE_host!10!CHECK_JMXMEM!40005!4248302272!5498760192!<
optional password for JMX authentication to ids-j>
        notifications_enabled           0
        }
```

See the Nagios documentation for a description of service definition format.

DirX Product Suite

The DirX product suite provides the basis for fully integrated identity and access management; it includes the following products, which can be ordered separately.



DirX Identity

DirX Identity provides a comprehensive, process-driven, customizable, cloud-enabled, scalable, and highly available identity management solution for businesses and organizations. It provides overarching, risk-based identity and access governance functionality seamlessly integrated with automated provisioning. Functionality includes lifecycle management for users and roles, cross-platform and rule-based real-time provisioning, web-based self-service functions for users, delegated administration, request workflows, access certification, password management, metadirectory as well as auditing and reporting functionality.



DirX Directory

DirX Directory provides a standards-compliant, high-performance, highly available, highly reliable, highly scalable, and secure LDAP and X.500 Directory Server and LDAP Proxy with very high linear scalability. DirX Directory can serve as an identity store for employees, customers, partners, subscribers, and other IoT entities. It can also serve as a provisioning, access management and metadirectory repository, to provide a single point of access to the information within disparate and heterogeneous directories available in an enterprise network or cloud environment for user management and provisioning.



DirX Access

DirX Access is a comprehensive, cloud-ready, scalable, and highly available access management solution providing policy- and risk-based authentication, authorization based on XACML and federation for Web applications and services. DirX Access delivers single sign-on, versatile authentication including FIDO, identity federation based on SAML, OAuth and OpenID Connect, just-in-time provisioning, entitlement management and policy enforcement for applications and services in the cloud or on-premises.



DirX Audit

DirX Audit provides auditors, security compliance officers and audit administrators with analytical insight and transparency for identity and access. Based on historical identity data and recorded events from the identity and access management processes, DirX Audit allows answering the “what, when, where, who and why” questions of user access and entitlements. DirX Audit features historical views and reports on identity data, a graphical dashboard with drill-down into individual events, an analysis view for filtering, evaluating, correlating, and reviewing of identity-related events and job management for report generation.

For more information: support.dirx.solutions/about

Eviden is a registered trademark © Copyright 2026, Atos Group – All rights reserved.

Legal remarks

On the account of certain regional limitations of sales rights and service availability, we cannot guarantee that all products included in this document are available through the Eviden sales organization worldwide. Availability and packaging may vary by country and is subject to change without prior notice. Some/All of the features and products described herein may not be available locally. The information in this document contains general technical descriptions of specifications and options as well as standard and optional features which do not always have to be present in individual cases. Eviden reserves the right to modify the design, packaging, specifications and options described herein without prior notice. Please contact your local Eviden sales representative for the most current information. Note: Any technical data contained in this document may vary within defined tolerances. Original images always lose a certain amount of detail when reproduced.