# Nirva DIALOGUE Service

Document Version: 1.13

# Table of Contents

# Overview

The DIALOGUE service is a NIRVA external service that allows the communication to the HP DIALOGUE document composition engine.

It provides functionality to simplify the deployment and production steps of a document composition project.

Through the NIRVA connectors, it also allows easy integration of document composition into external applications or environments.

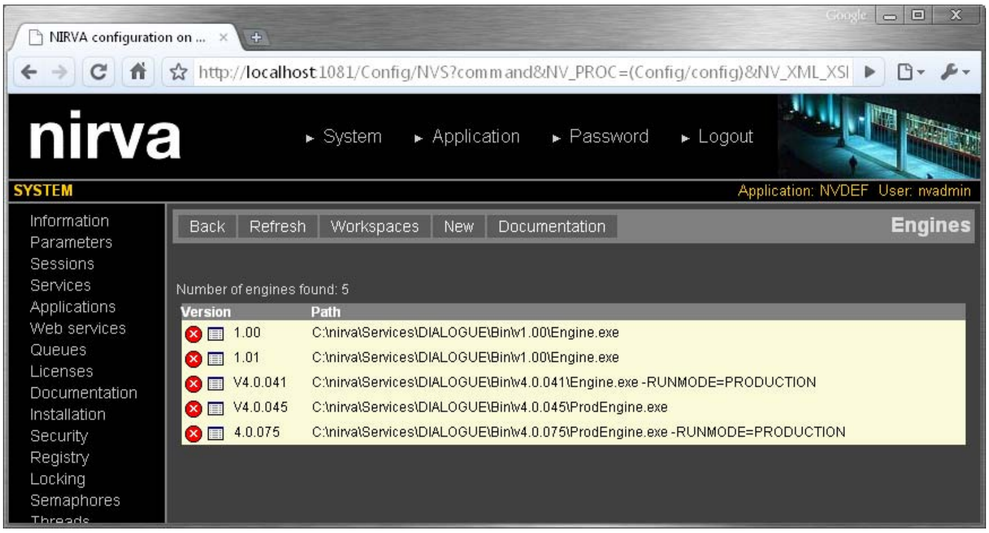
The NIRVA DIALOGUE service manages:

- DIALOGUE engine versions.
- Production workspaces.
- DIALOGUE package repositories.
- Document types.
- Document composition.

## Versions

The service provides the management of DIALOGUE engine versions. DIALOGUE engines are executables that realize the composition step according to parameters and information provided in one or several input files and in a configuration file called “package”. The package file is generated by the DIALOGUE tools.

A DIALOGUE package file is often linked to a specific engine version. That’s why it’s important to manage the DIALOGUE engine versions for production.



Number of engines found: 5

| Version | Path |
|---|---|
| 1.00 | C:\nirva\Services\DIALOGUE\Bin\v1.00\Engine.exe |
| 1.01 | C:\nirva\Services\DIALOGUE\Bin\v1.00\Engine.exe |
| V4.0.041 | C:\nirva\Services\DIALOGUE\Bin\v4.0.041\Engine.exe -RUNMODE=PRODUCTION |
| V4.0.045 | C:\nirva\Services\DIALOGUE\Bin\v4.0.045\ProdEngine.exe |
| 4.0.075 | C:\nirva\Services\DIALOGUE\Bin\v4.0.075\ProdEngine.exe -RUNMODE=PRODUCTION |

This DIALOGUE NIRVA service does not provide the DIALOGUE engines. These engines must be installed on the NIRVA computer. The service only requires the pathnames of these DIALOGUE engines.

# Workspaces

A workspace represents an environment of a given application. For example, for a given application or set of applications, it is possible to create three environments: development, test and production. These workspaces are named objects that include the composition package and document type repositories. The number of workspaces is not limited.

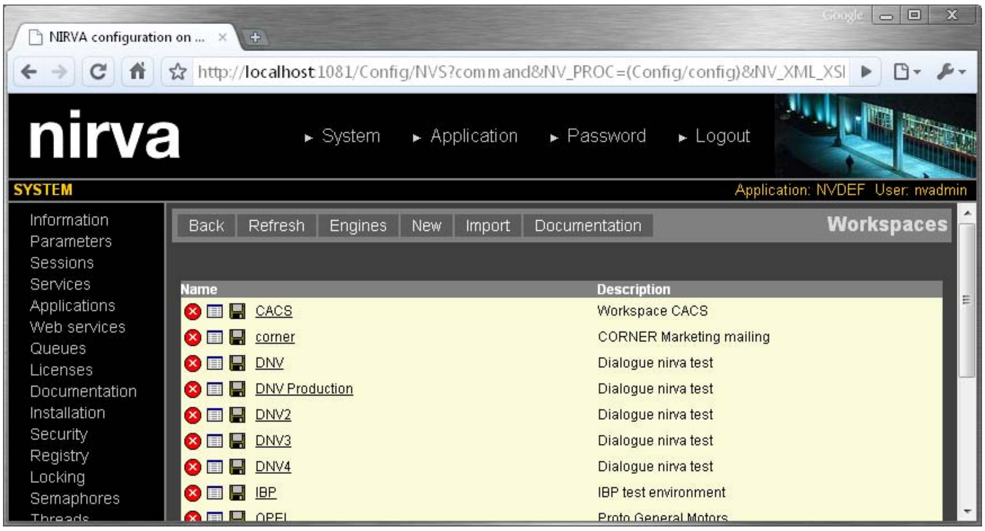
The workspace can be exported and installed across platforms even if the target platforms are not using the same operating system. One can for example create a workspace under a windows platform and install it on an AIX platform.

The workspace greatly facilitates the deployment of document production applications.

When a composition run is launched, the NIRVA user has to specify the name of the workspace to use.

The workspace feature is very useful for separating production environments from testing ones or in the case of an outsourcing NIRVA platform.

When the dialogue service is installed, the list of workspaces is available in the Nirva configuration tool, in the System/Services/Dialogue menu.

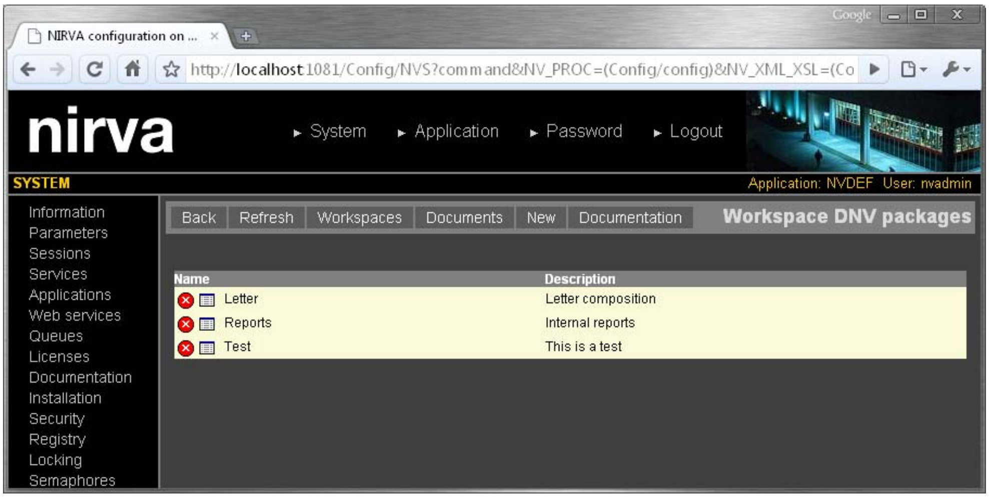


| Name | Description |
|---|---|
| CACS | Workspace CACS |
| corner | CORNER Marketing mailing |
| DNV | Dialogue nirva test |
| DNV Production | Dialogue nirva test |
| DNV2 | Dialogue nirva test |
| DNV3 | Dialogue nirva test |
| DNV4 | Dialogue nirva test |
| IBP | IBP test environment |
| OPEL | Proto General Motors |

# Packages

The DIALOGUE service provides a repository for DIALOGUE packages. A DIALOGUE package is a binary file created with DIALOGUE tools that contains all the necessary configuration information for the DIALOGUE engine to run the composition, for one or several document types.

The NIRVA DIALOGUE service allows hiding the packages to the final user who is just interested with document types.

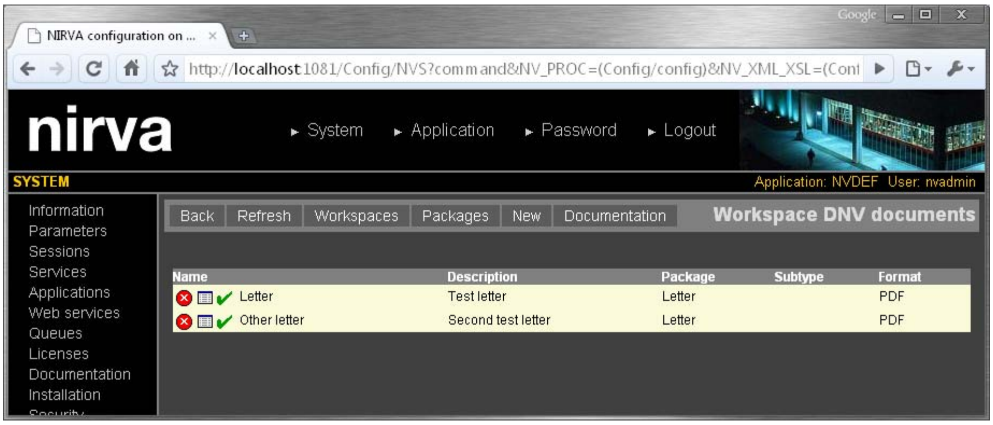


| Name | Description |
|---|---|
| Letter | Letter composition |
| Reports | Internal reports |
| Test | This is a test |

# Documents

The DIALOGUE NIRVA service provides a repository for the document types to compose.

The document type is the only thing that the user of the service is interested with. When the composition is requested to NIRVA, NIRVA retrieves the corresponding DIALOGUE package in its package repository and runs the composition.

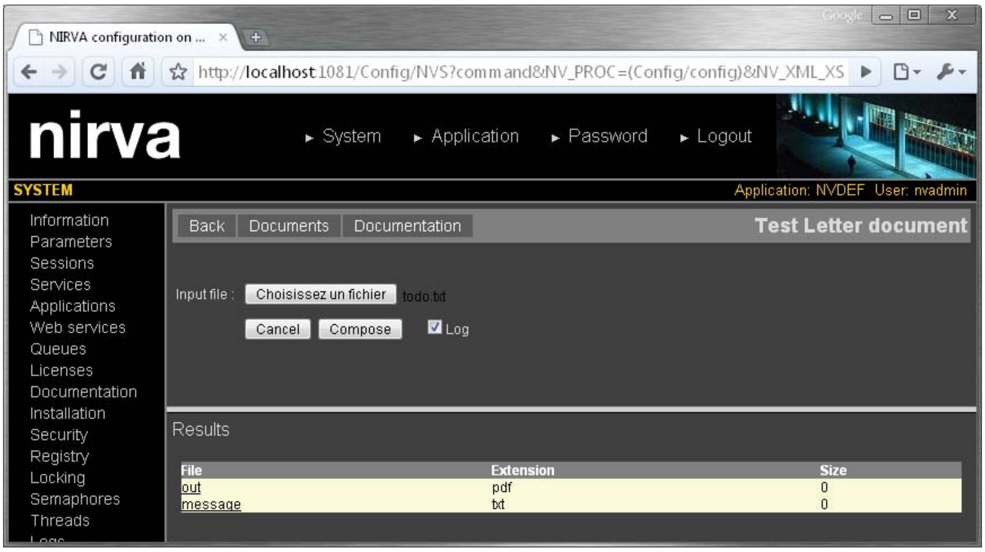


Workspace DNV documents

| Name | Description | Package | Subtype | Format |
|---|---|---|---|---|
| Letter | Test letter | Letter | | PDF |
| Other letter | Second test letter | Letter | | PDF |

## Composition

NIRVA can start the composition of a document with a single command. The composition run can be based on one or several entry files. The format of these entry files must be recognized by DIALOGUE. The composition run will then be able to deliver one ore several output files. For example, a document can be defined to be produced in PDF format, after the reception of two input files: one containing data and another containing some extra parameters (logo display, type of chart, etc.). The input and output files are NIRVA file objects. The NIRVA DIALOGUE service manages the mapping between NIRVA object names and DIALOGUE file names.



Test Letter document

Input file : Choisissez un fichier todo.txt

☑ Log

Results

| File | Extension | Size |
|---|---|---|
| out | pdf | 0 |
| message | txt | 0 |

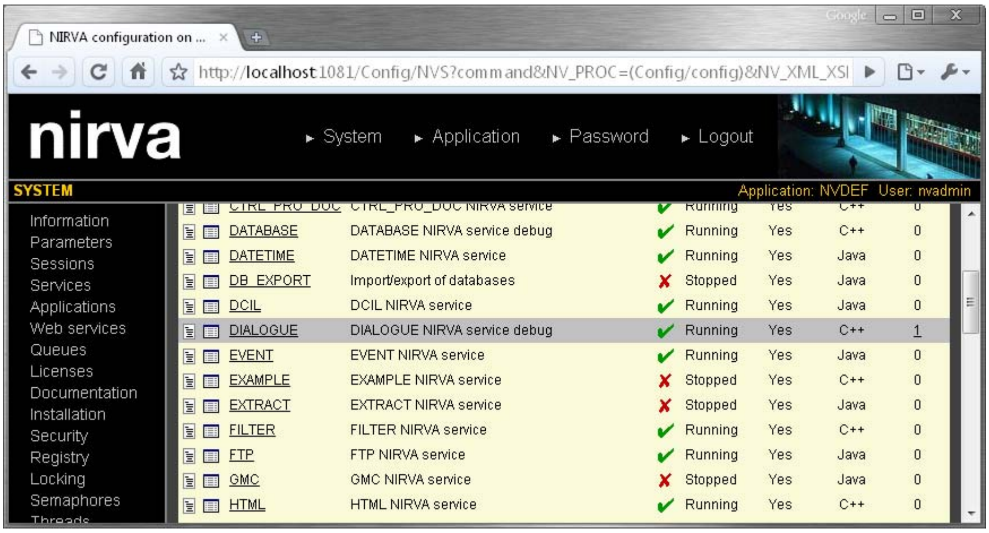
# Installation

The DIALOGUE service is a standard NIRVA package and can thus be installed like any other NIRVA service, using the NIRVA configuration web pages. Please refer to the NIRVA configuration chapter in the NIRVA user's guide for further information.

# Configuration

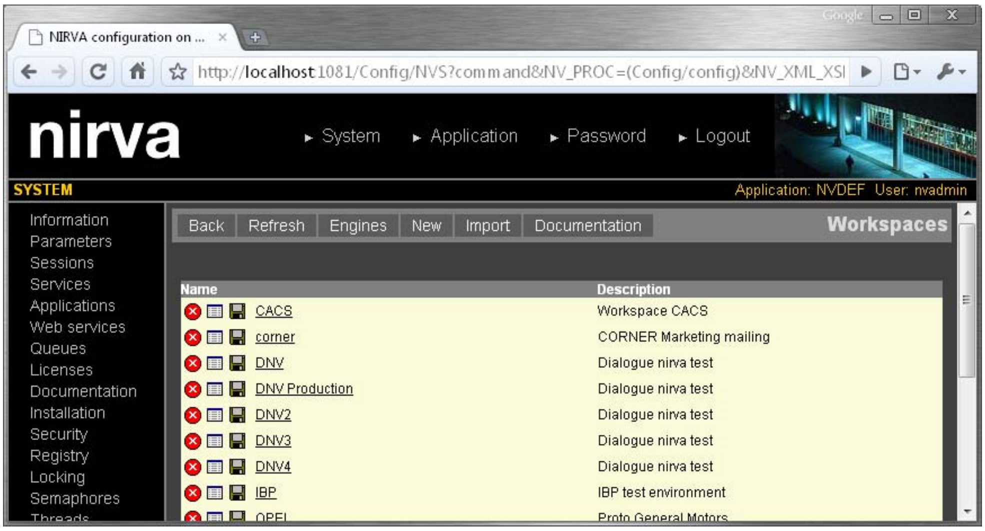
The DIALOGUE service configuration is entirely dynamic and available from a web browser.

The configuration of the DIALOGUE service is accessible directly from the main list of services of the NIRVA configuration tool:



| | | | | | |
|---|---|---|---|---|---|
| CTRL_PRO_DOC | CTRL_PRO_DOC NIRVA service | ✓ Running | Yes | C++ | 0 |
| DATABASE | DATABASE NIRVA service debug | ✓ Running | Yes | C++ | 0 |
| DATETIME | DATETIME NIRVA service | ✓ Running | Yes | Java | 0 |
| DB_EXPORT | Import/export of databases | ✗ Stopped | Yes | Java | 0 |
| DCIL | DCIL NIRVA service | ✓ Running | Yes | Java | 0 |
| DIALOGUE | DIALOGUE NIRVA service debug | ✓ Running | Yes | C++ | 1 |
| EVENT | EVENT NIRVA service | ✓ Running | Yes | Java | 0 |
| EXAMPLE | EXAMPLE NIRVA service | ✗ Stopped | Yes | C++ | 0 |
| EXTRACT | EXTRACT NIRVA service | ✗ Stopped | Yes | Java | 0 |
| FILTER | FILTER NIRVA service | ✓ Running | Yes | C++ | 0 |
| FTP | FTP NIRVA service | ✓ Running | Yes | Java | 0 |
| GMC | GMC NIRVA service | ✗ Stopped | Yes | Java | 0 |
| HTML | HTML NIRVA service | ✓ Running | Yes | C++ | 0 |

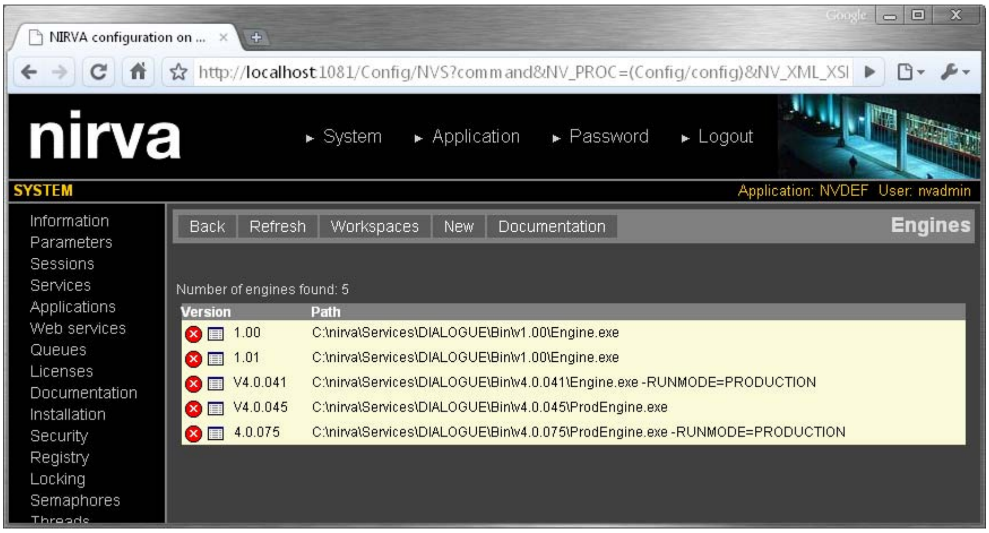
When choosing the configuration icon of the DIALOGUE service (▤), it starts the DIALOGUE service configuration by displaying the list of DIALOGUE workspaces:



| Name | Description |
|---|---|
| CACS | Workspace CACS |
| corner | CORNER Marketing mailing |
| DNV | Dialogue nirva test |
| DNV Production | Dialogue nirva test |
| DNV2 | Dialogue nirva test |
| DNV3 | Dialogue nirva test |
| DNV4 | Dialogue nirva test |
| IBP | IBP test environment |
| OPEL | Proto General Motors |

## Engine versions

The engine management is available by clicking on the “Engines” button from the workspace list page.



Number of engines found: 5

| Version | Path |
|---|---|
| 1.00 | C:\nirva\Services\DIALOGUE\Bin\v1.00\Engine.exe |
| 1.01 | C:\nirva\Services\DIALOGUE\Bin\v1.00\Engine.exe |
| V4.0.041 | C:\nirva\Services\DIALOGUE\Bin\v4.0.041\Engine.exe -RUNMODE=PRODUCTION |
| V4.0.045 | C:\nirva\Services\DIALOGUE\Bin\v4.0.045\ProdEngine.exe |
| 4.0.075 | C:\nirva\Services\DIALOGUE\Bin\v4.0.075\ProdEngine.exe -RUNMODE=PRODUCTION |

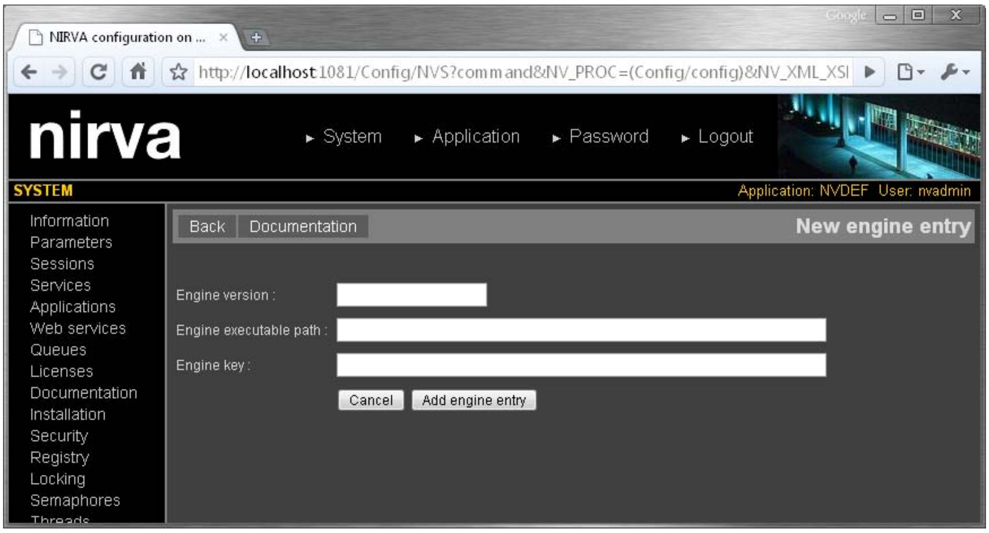
This window displays the list of the mounted versions of DIALOGUE engines. The DIALOGUE service doesn't install any DIALOGUE engine, but refers to the installed ones.

A DIALOGUE engine is a single executable named "Engine.exe" for Windows systems and "Engine" for UNIX platforms. A file named "MessageSymbolics.h" must also be present in the directory of the engine executable.

The “Workspaces” button can be used to come back to the workspaces list.

### Mounting a new engine version

Mounting a new engine is done by clicking the button “New”. It displays the following screen:



Engine version :

Engine executable path :

Engine key :

The engine version is mandatory. It’s case independent. The version number that the NIRVA DIALOGUE service uses to specify an engine is independent of the real DIALOGUE version number. It is though recommended to choose a fixed-length for the version string and to use figures and dots only. This way, it is possible to force NIRVA to automatically choose the most recent engine when running a composition. NIRVA defines the newest version after an alphabetical sort.

The “Engine executable path” parameter must point to the DIALOGUE engine command line. One can add some DIALOGUE specific extra parameters to the command line. Generally, this is not necessary because the complete command line is built by the Nirva service, at composition time. NIRVA processes this way: it takes the command line as defined here above in the configuration tool and adds to it the necessary parameters pointing to input and output files.

The “Engine key” parameter is optional: it is the license key of the DIALOGUE engine.

## Removing an engine entry

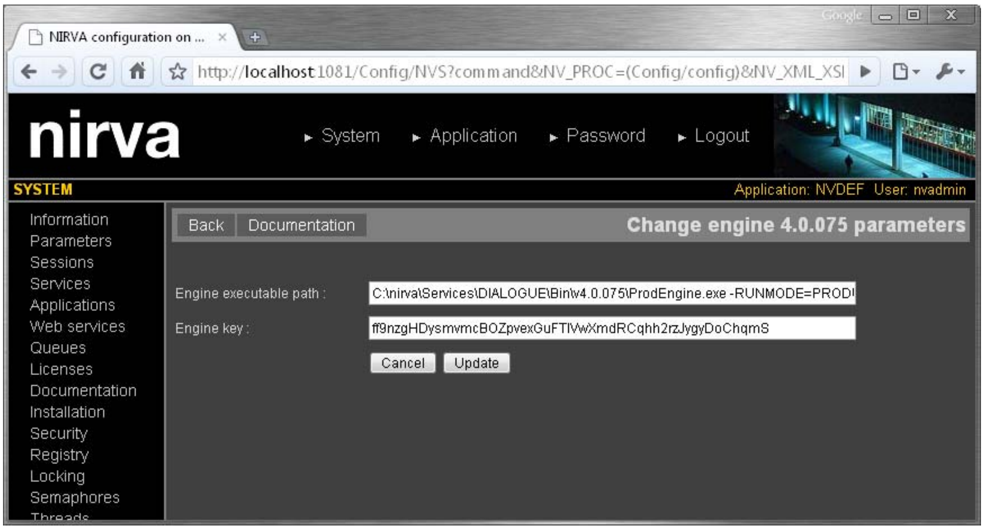
Removing an engine entry is quite simple. In the engine list, select the ✖ icon next to the engine version. The configuration tool will request a confirmation and try to remove the engine: for doing that it will internally wait for 10 seconds for the engine to be free. Only free engines can be removed.

If the engine cannot be removed within 10 seconds, an error message is displayed. In this case, the order to remove must be given again or a manual ENGINE:REMOVE command must be sent with a WAIT parameter set to a value greater than 10. See the command reference chapter for further information about the ENGINE:REMOVE command.

The removal only concerns the entry in the internal service list. The command doesn’t physically remove any DIALOGUE file.

## Changing engine entry parameters

To change the engine entry parameters, just select the ▤ icon near the engine version in the engine list. It displays the following window:



Change engine 4.0.075 parameters

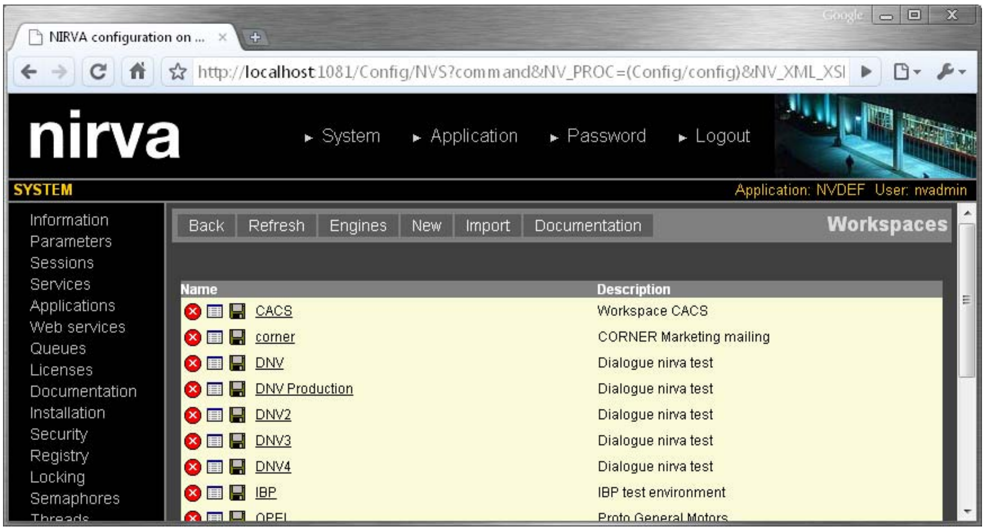
Engine executable path : C:\nirva\Services\DIALOGUE\Bin\v4.0.075\ProdEngine.exe -RUNMODE=PROD

Engine key : ff9nzgHDysmvmcBOZpvexGuFTlVwXmdRCqhh2rzJygyDoChqmS

The engine version cannot be changed from this window. The only way to change the engine version is to remove the engine entry and to create it again.

## Workspaces

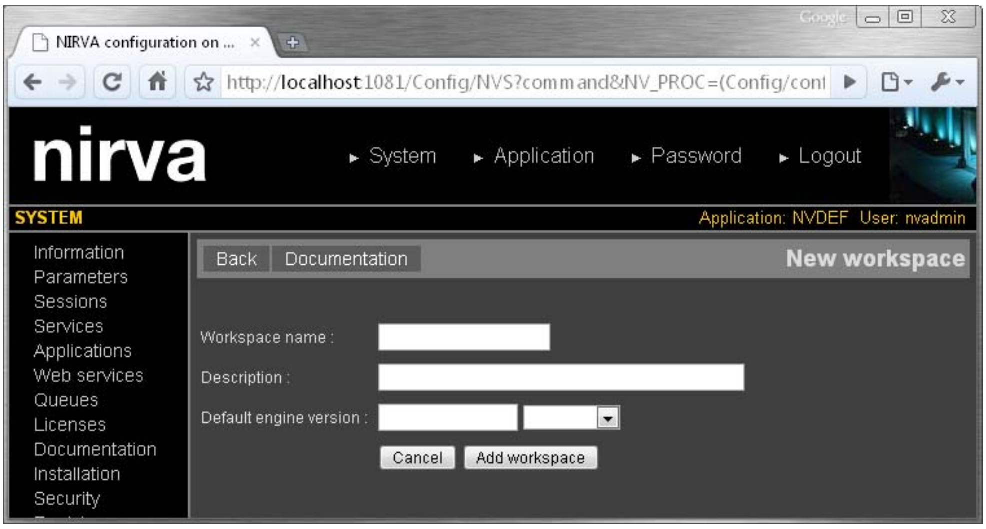
The list of available workspaces is displayed automatically when entering the configuration of the DIALOGUE service:



| Name | Description |
|---|---|
| CACS | Workspace CACS |
| corner | CORNER Marketing mailing |
| DNV | Dialogue nirva test |
| DNV Production | Dialogue nirva test |
| DNV2 | Dialogue nirva test |
| DNV3 | Dialogue nirva test |
| DNV4 | Dialogue nirva test |
| IBP | IBP test environment |
| OPEL | Proto General Motors |

### Creating a new workspace

A new workspace can be created by clicking on the button “New”. It displays the following screen:



Workspace name :

Description :

Default engine version :

The workspace name is mandatory. It is case independent and can contain spaces.

The “Default engine version” parameter is optional. It gives the default DIALOGUE engine version for this workspace. It must be an installed DIALOGUE engine version.

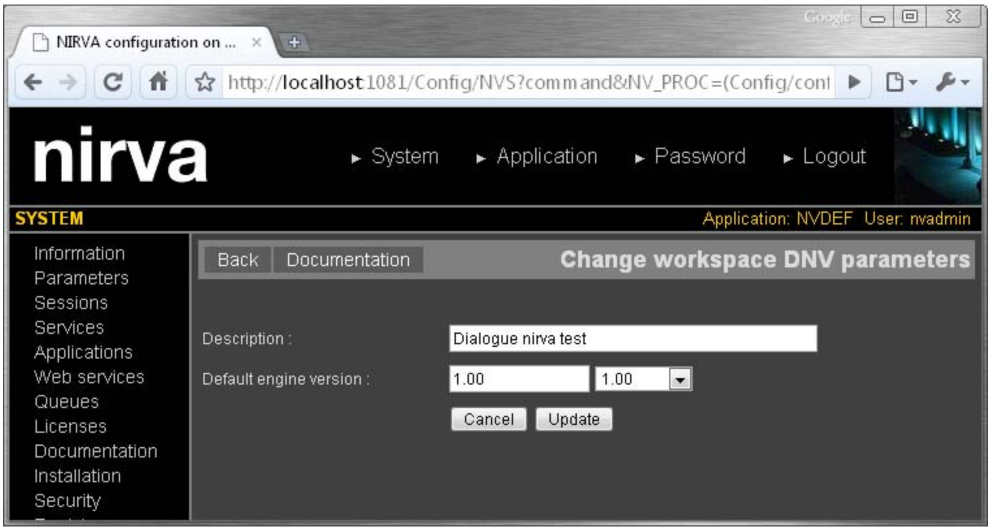
### Removing a workspace

Removing a workspace is quite simple. Select the ⊗ icon near the workspace name in the workspace list. The configuration tool will request a confirmation and try to remove the workspace: for doing that it will internally wait for 10 seconds for the workspace to be free. Only free workspaces can be removed.

If the workspace cannot be removed within 10 seconds, an error message is displayed. In this case, the order to remove the workspace must be given again or a manual WORKSPACE:REMOVE command must be sent with a WAIT parameter set to a value greater than 10. See the command reference chapter for further information about the WORKSPACE:REMOVE command.

## Changing workspace parameters

To change the workspace parameters, select the ▤ icon next to the workspace name in the workspace list. It displays the following window:



Change workspace DNV parameters

Description : Dialogue nirva test

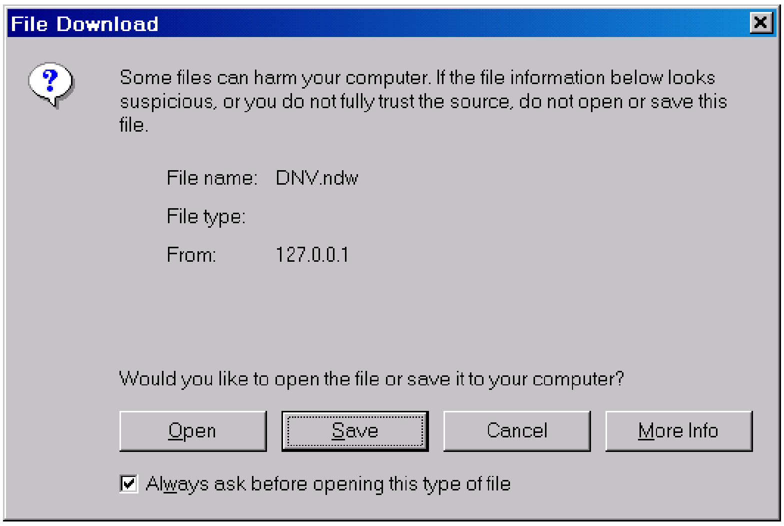
Default engine version : 1.00 1.00

The only way to change the workspace name is to remove the workspace and to create it again.

## Exporting a workspace

The workspace export command is used for workspace deployment. An exported workspace is a single file containing the entire workspace including document definitions and DIALOGUE packages.

Exporting a workspace is done by selecting the 💾 icon next to the workspace name in the workspace list. You are then prompted to locally download the workspace:



Some files can harm your computer. If the file information below looks suspicious, or you do not fully trust the source, do not open or save this file.

File name: DNV.ndw

File type:

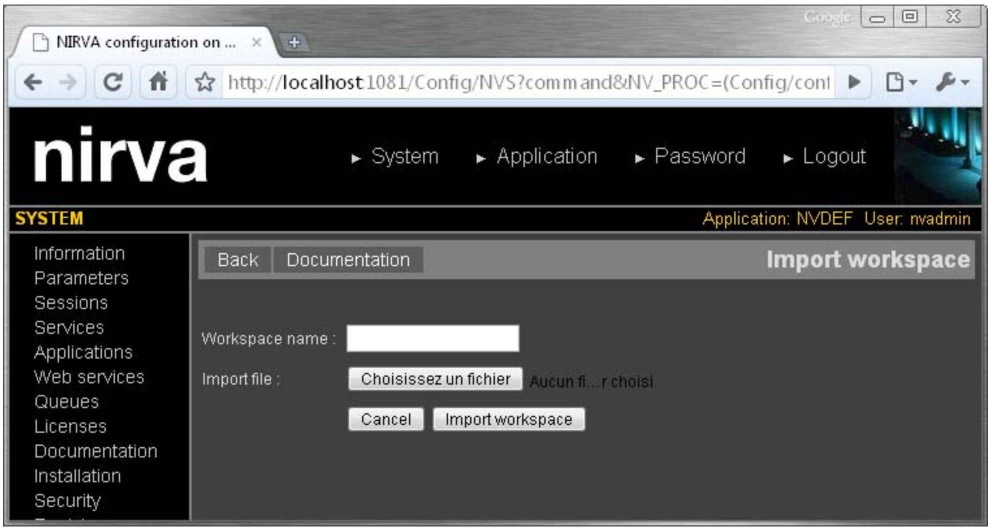
From: 127.0.0.1

Would you like to open the file or save it to your computer?

☑ Always ask before opening this type of file

### Importing a workspace

The installation of a workspace can be done by importing it. To do that, click on the “Import” button above the workspace list. It displays the following window:



Workspace name :

Import file : Aucun fi...r choisi

The workspace name is generally taken directly from the workspace imported file but it can be changed. This is useful to install another version of the same workspace for testing purpose for instance.

The import file must point to your local workspace file. This can only be a file that has previously been created by the workspace export command.

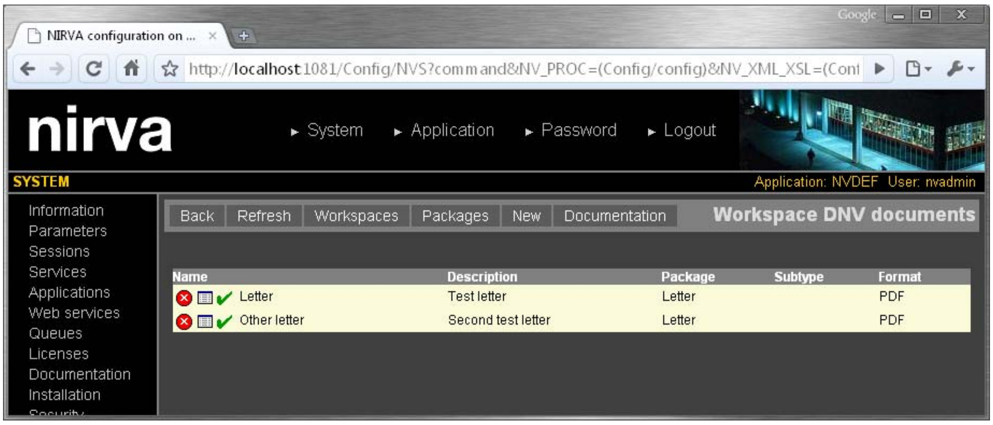
### Editing a workspace

By clicking on the workspace name, one can edit the workspace. It gives access to the defined documents and packages of the workspace.

## Documents

The documents are defined at workspace level. All the documents to be used later in document composition must be defined here.

This then displays the following screen:



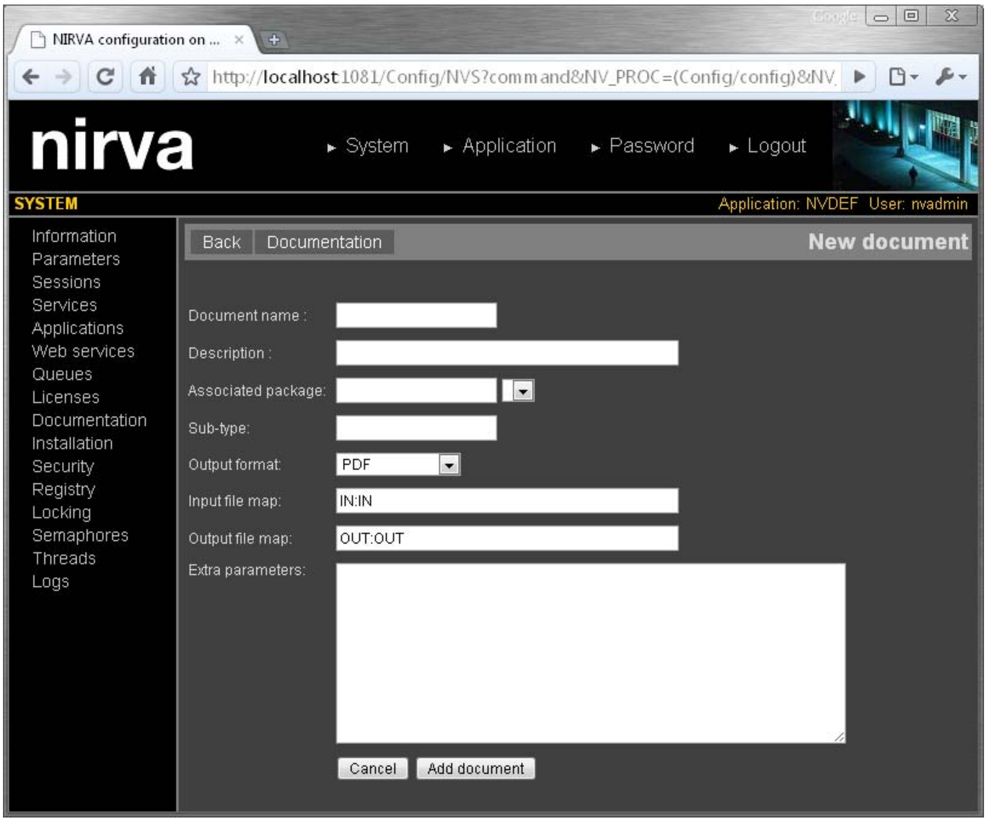
| Name | Description | Package | Subtype | Format |
|---|---|---|---|---|
| Letter | Test letter | Letter | | PDF |
| Other letter | Second test letter | Letter | | PDF |

This is the list of current defined documents. For each document the list gives the following information:

- *Name* is the document name. It is case independent.
- *Description* is an optional description of the document.
- *Package* is the name of the DIALOGUE package for the document.
- *Subtype* is the eventual document type at package level. It can be used when a single DIALOGUE package manages several document types.
- *Format* is the default document output format.

## Creating a new document

Creating a new document is done by clicking on the button “New” on the document list screen. This displays the following screen:



Document name :

Description :

Associated package:

Sub-type:

Output format: PDF

Input file map: IN:IN

Output file map: OUT:OUT

Extra parameters:

The document name is case independent. If the user attempts to create a document with the name of an existing one, the command fails. The document name is mandatory.

The document description is a simple string that describes the document. It is not mandatory.

The associated package is the name of the DIALOGUE package that processes the document. The package must have been previously defined in the package list. Several package names can be defined (multi packages feature): they must be separated by a semicolon character (;).

Sub-type can be used when a DIALOGUE package contains the definition of several documents. The subtype is a string transmitted to the DIALOGUE engine by a specific file map named “DOCTYPE”. At composition time, the DIALOGUE service creates the XML file containing the subtype and sends it to the engine as the “DOCTYPE” file map. The XML file has a single element named “DOCTYPE” that contains the subtype string. Here is an example of such a file:

```
<DOCTYPE>Mydoctype</DOCTYPE>
```

Where *Mydoctype* is the string given in the document subtype.

Output format is the default output format for the document.

Input file map is a collection of strings separated by a semicolon character that gives the mapping between NIRVA file objects and DIALOGUE input file maps. Each string has the following format: *NirvaObject*:*DialogueFileMap* where *NirvaObject* is the name of the NIRVA file object containing the input file and *DialogueFileMap* is the name of the DIALOGUE file map for the input. If the *NirvaObject* is given alone, the DIALOGUE file map is supposed to have the same name. The “DOCTYPE” file map name is reserved for transmitting the subtype to DIALOGUE and cannot be used.

If *DialogueFileMap* is set to “NV_DYNAMIC_PACKAGE”, Nirva considers the input file as a dynamic package and sends the file to DIALOGUE as a package instead of as a file map.

Output file map is a collection of strings separated by a semicolon character that gives the mapping between NIRVA file objects and DIALOGUE output file maps. Each string has the following format: *NirvaObject*:*DialogueFileMap*(*OutputFormat*) where *NirvaObject* is the name of the NIRVA file object containing the output file, *DialogueFileMap* is the name of the DIALOGUE file map for the output and *OutputFormat* is the output file format.. If the *NirvaObject* is given alone, the DIALOGUE file map is supposed to have the same name. If the *OutputFormat* is omitted, NIRVA uses the default output format defined for this document. The output formats can be “PDF”, “AFP”, “HTML”, “RTF”, “TIFF”, “PCL”, “XML”, “TXT”, “METACODE”, “POSTSCRIPT”, “VPS”, “IJPDS”, “VIPP”, “PPML” or “3211”. The “DOCTYPE” file map name is reserved for transmitting the subtype to DIALOGUE and cannot be used.

The “NV_NO_FILEMAP” file map has a special meaning. It must be used with a dialogue engine version that doesn’t have the High Volume Print Mail option. At this time, the service is using the –outputfile option on the Dialogue command line instead of a –filemap option.

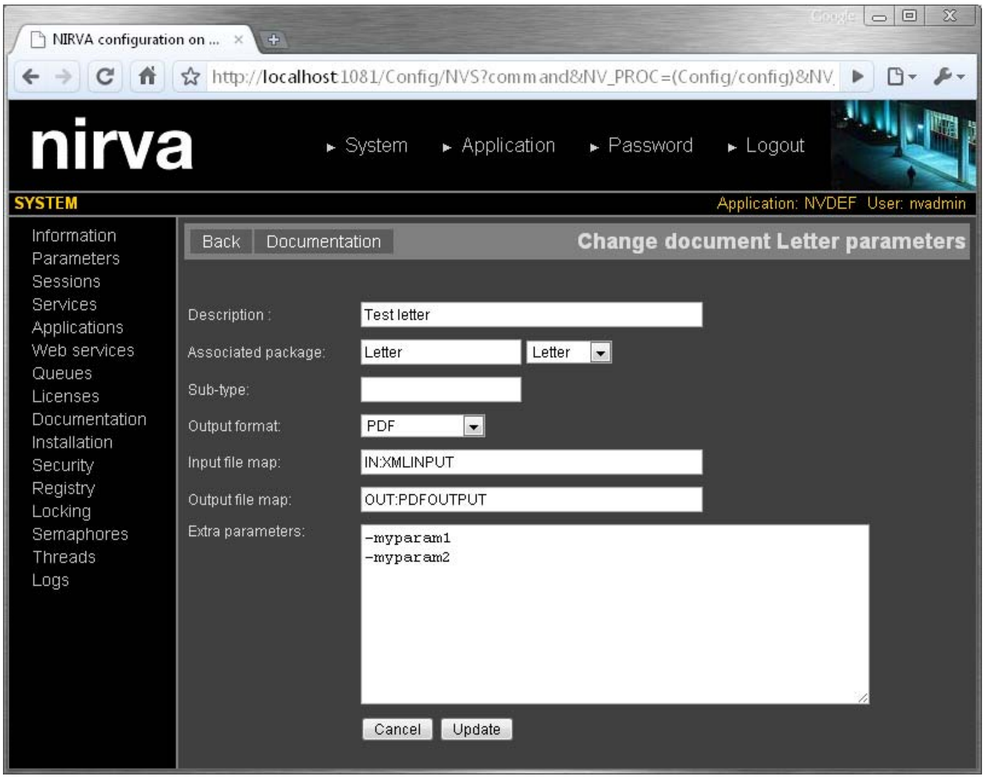
Extra parameters are document level parameters added to the dialogue command line (ex. Variables). There must have one parameter per line.

### Removing a document

Removing a document is quite simple. Select the ⊗ icon next to the document name in the document list. The configuration tool will request a confirmation and try to remove the document (if not in use).

### Changing document parameters

To change the document parameters, just select the ▤ icon next to the document name in the document list. It displays the following window:



**Change document Letter parameters**

Description : Test letter

Associated package: Letter Letter

Sub-type:

Output format: PDF

Input file map: IN:XMLINPUT

Output file map: OUT:PDFOUTPUT

Extra parameters:

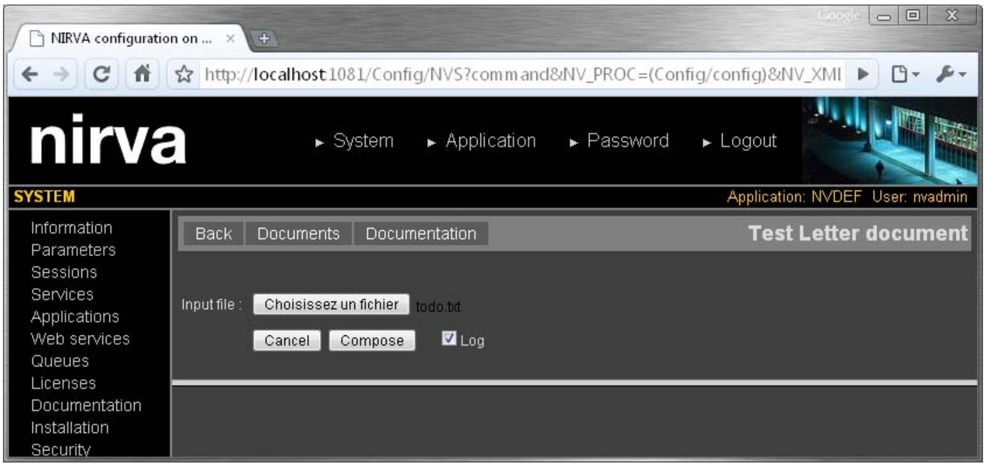
```
-myparam1
-myparam2
```

All the document parameters can be changed except its name.

Please refer to the “Creating a new document” chapter for a description of document parameters.

## Testing a document

It’s possible to test the composition of a document by pressing the ✓ icon next to its name from the document list. The displayed window depends on the file maps defined for the document. There is one requested input file for each defined input file map. Here is an example with a single input file object named “IN”:



**Test Letter document**

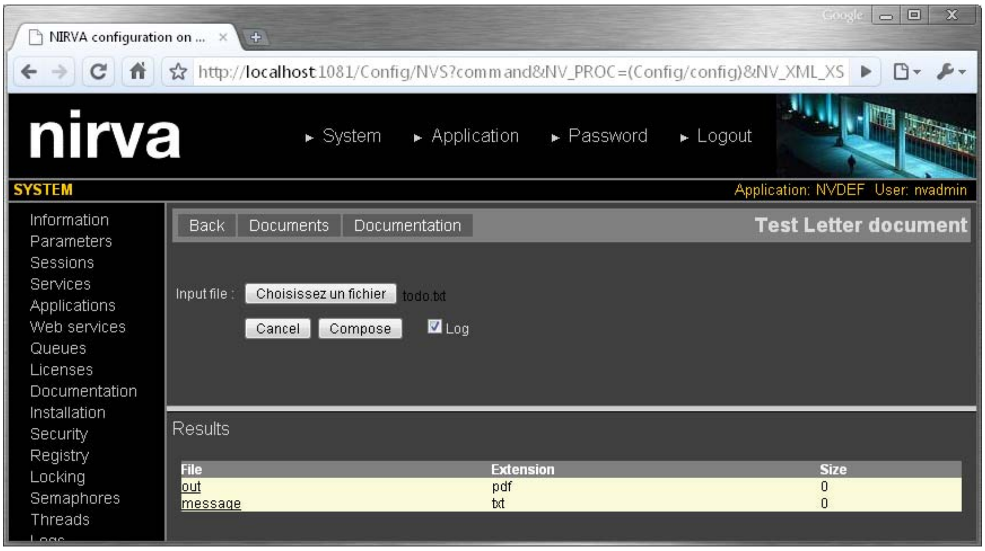
Input file : todo.txt

☑ Log

When the input file object is named “IN”, Nirva displays the text “Input file” instead of “IN”.

For each input file, one must give a local file name for tests and press the “Compose” button to test the composition.

The “Log” check box also allows displaying the DIALOGUE message file. If the composition is successful, NIRVA displays the list of output files:



http://localhost:1081/Config/NVS?command&NV_PROC=(Config/config)&NV_XML_XS

Application: NVDEF User: nvadmin

Test Letter document

Input file : todo.txt

☑ Log

Results

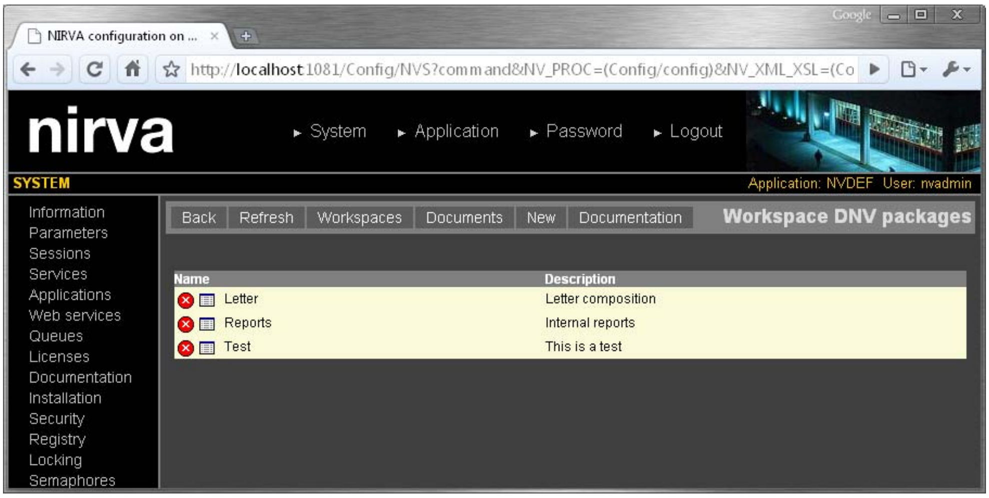
| File | Extension | Size |
|---|---|---|
| out | pdf | 0 |
| message | txt | 0 |

The output files can then be displayed (or downloaded) by clicking on their name. The name of the output file is the name of the created Nirva object (as defined in the output file map for this document).

## Packages

The packages are defined at workspace level. NIRVA maintains a DIALOGUE package repository for each workspace.

The package list is available from the document list by clicking on the “Packages” button. It displays the following screen:



http://localhost:1081/Config/NVS?command&NV_PROC=(Config/config)&NV_XML_XSL=(Co

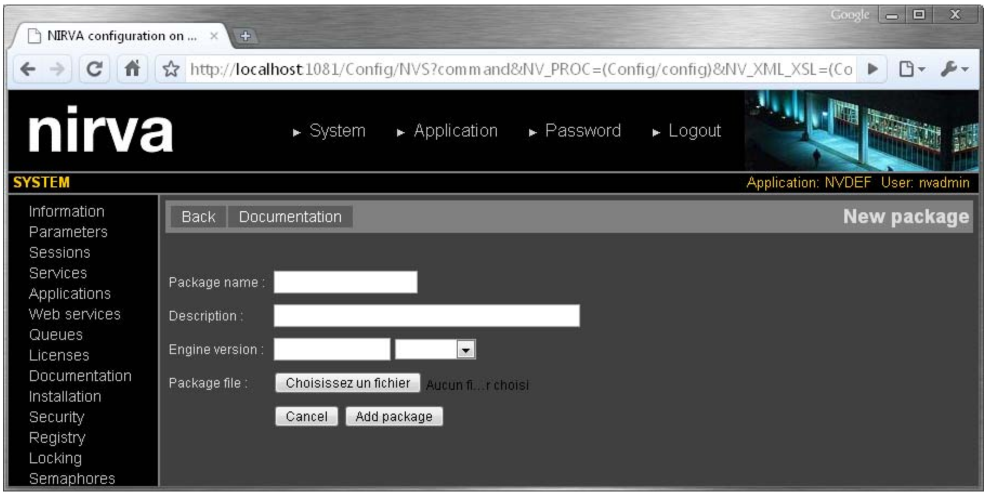
Application: NVDEF User: nvadmin

Workspace DNV packages

| Name | Description |
|---|---|
| Letter | Letter composition |
| Reports | Internal reports |
| Test | This is a test |

### Creating a new package

Creating a new package can be done by clicking on the button “New”. It displays the following screen:



The package name is mandatory. It is case independent and can contain spaces.

The “Engine version” parameter is optional. It gives the DIALOGUE engine version to use for this package. It must be an installed DIALOGUE engine version. If not given, NIRVA will use the default engine version of the workspace or will try to get the last one.

“Package file” must point to the real DIALOGUE package file. NIRVA then uploads it in its package repository. It can be done at package creation time but later as well.

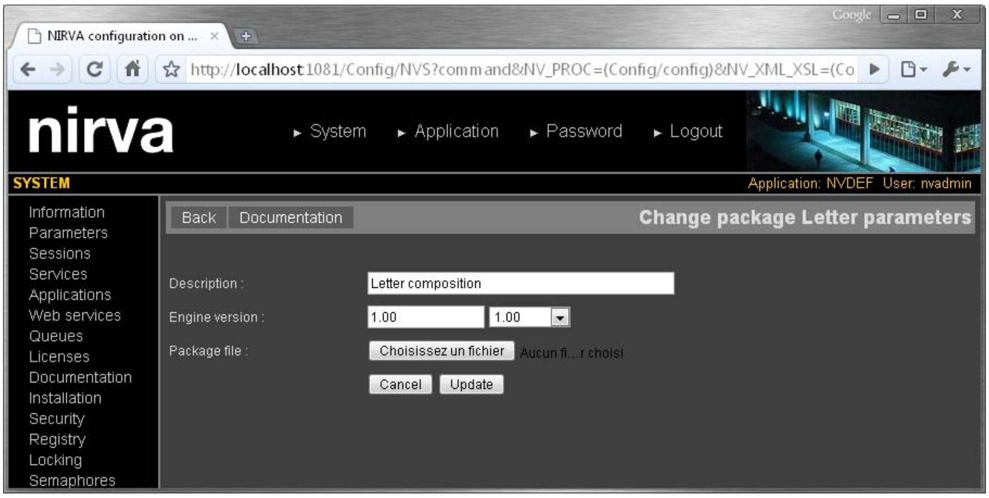
## Removing a package

Removing a package is quite simple. Select the ⊗ icon next to the package name in the package list. The configuration tool will request a confirmation and try to remove the package: For doing that it will internally wait for 10 seconds for the package to be free. Only free packages can be removed.

If the package cannot be removed within 10 seconds, an error message is displayed. In this case, the order to remove must be given again or a manual PACKAGE:REMOVE command must be sent with a WAIT parameter set to a value greater than 10. Please refer to the command reference chapter for further information about the PACKAGE:REMOVE command.

## Changing package parameters

To change the package parameters, select the ▤ icon next to the package name in the package list. It displays the following window:



Application: NVDEF User: nvadmin

Change package Letter parameters

Description : Letter composition

Engine version : 1.00 1.00

Package file :

The package name cannot be changed this way. The only way to change the package name is to remove the package and to create it again.

The package file will be updated only if a new file is given.

# Reference

This chapter gives the complete reference of all the DIALOGUE service commands.

## Classes

Here are the available DIALOGUE service classes:

| | |
|---|---|
| DIALOGUE | Main class. |
| ENGINE | DIALOGUE engine versions management. |
| WORKSPACE | Workspace management. |
| PACKAGE | DIALOGUE packages repository management. |
| DOCUMENT | Document management. |

## Error codes

### COMMAND Class

| Value | Description |
|---|---|
| 101 | Invalid command |

### DOCUMENT Class

| Value | Description |
|---|---|
| 101 | Document not found |
| 102 | The document already exists |
| 103 | Cannot create the document |

| Value | Description |
|---|---|
| 104 | No document name given |
| 105 | Input file missing |
| 106 | No output file |
| 107 | Cannot set the document subtype for dialogue |
| 108 | Error in composition |

## ENGINE Class

| Value | Description |
|---|---|
| 101 | Cannot save the engine list |
| 102 | Engine not found |
| 103 | The engine entry already exists |
| 104 | Cannot create the engine entry |
| 105 | No engine version given |
| 106 | The engine is in use |
| 107 | Cannot get Dialogue engine |

## PACKAGE Class

| Value | Description |
|---|---|
| 101 | Package not found |
| 102 | The package already exists |
| 103 | Cannot create the package |
| 104 | No package name given |
| 105 | Cannot set the package file |
| 106 | Cannot get the package file |

## WORKSPACE Class

| Value | Description |
|---|---|
| 101 | Cannot save the workspace list |
| 102 | Workspace not found |
| 103 | The workspace already exists |
| 104 | Cannot create the workspace |
| 105 | No workspace name given |

| Value | Description |
|---|---|
| 106 | The workspace is in use |
| 107 | Cannot save workspace information to registry |
| 108 | Cannot export workspace to file |
| 109 | Cannot import workspace from file |

# Permissions

| Name | Description |
|---|---|
| ADMIN | Administration (update mode) |

# Commands

For each command, the reference gives the command name, the sources from which the command may be used, the command description, the eventual command permissions, the parameter list and the eventual list of objects created by the command.

The parameters described in this chapter are command specific parameters. For general parameters, please refer to the Nirva command syntax chapter.

The available command sources are:

- Client for all Nirva client interfaces including Nirva client library (nvc).
- Web for commands from a web browser.
- Procedure for commands from a Nirva procedure.
- Service for commands from service to service.

## DIALOGUE class

This is the standard service class. It only provides just one command, for testing purpose.

### NOP

DIALOGUE:DIALOGUE:NOP

| Source | Use Input Container | Use Output Container |
|---|---|---|

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | No |

### Description

This command does not do anything but allows to test that the DIALOGUE service is online and answers correctly.

If the service is not online, this command returns an error.

### Parameters

None

## DOCUMENT class

The DOCUMENT class provides commands to manage the document repositories and to launch the composition.

The document type is the only matter that the user of the service is interested with. When running the composition, a user gives the name of the workspace and the document type to compose to NIRVA. NIRVA gets the corresponding DIALOGUE package in its package repository and does the composition.

### COMPOSE

DIALOGUE:DOCUMENT:COMPOSE

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | Yes | Yes |

### Description

This command does the composition of a document by launching the DIALOGUE engine.

The purpose of the NIRVA DIALOGUE service is to simplify the composition step without having to take care about DIALOGUE package and engines.

For doing that, the COMPOSE command can be used by giving the following parameters only: the input files, the workspace name and the document type name to compose.

The COMPOSE command creates output file objects in the output container. The object names are defined by the output file maps of the document. For example, if the output file map has been defined as OUT:OUTPDF(PDF);OUT2:OUTAFP(AFP), the command will generate two file objects named “OUT” and “OUT2”. The attached file for these file objects have an extension corresponding to their defined format.

Parameters

*WORKSPACE* Workspace name. This parameter is mandatory.

*DOCUMENT* Document name. This parameter is mandatory.

*ENGINE* Engine version. This parameter can be given to specify a DIALOGUE engine version to use. If this parameter is not provided, NIRVA gets the engine version from the document associated package or, if not found, from the workspace default engine version parameter or, if not found, tries to evaluate the last engine version by alphabetically comparing the engine versions.

*File maps* All the input file maps defined in the document parameters must be given as NIRVA file objects in the input container. For example, if the file map IN:INPUT has been defined, the IN file object must exist in the input container.

*LOG* Keeps DIALOGUE log information file or not. When this parameter is set to “YES”, NIRVA creates an output file object named “MESSAGE” in the output container that gives log information about the DIALOGUE process.

*CREATE* Create mode. If one of the output file objects already exists, NIRVA doesn’t create it again but changes the content of its associated file. One can force the object creation by setting the CREATE parameter to “YES”. The default is “NO”.

*DIRECTORY* By default, NIRVA creates the output file objects in the application work directory. It can be changed by setting this parameter to the name of a directory.

*EXTENSION* The created object file extensions are automatically set by the command following the required output formats. It can be overridden by giving a specific extension in this parameter. At this time, if there is more than one output file, all the output files will have the given extension. If the dot character is omitted in the EXTENSION parameter, Nirva adds it.

*PREFIX* The PREFIX parameter is used in the file name when NIRVA creates the file objects.

*SUFFIX* The SUFFIX parameter is used in the file name when NIRVA creates the file objects.

*ERROR_NOSIZE* By default, the command doesn’t produce any error if one of the defined output file objects is empty. It can be changed by setting the ERROR_NOSIZE parameter to “YES”. In this case, if any of the output files is empty, the command returns an error.

*EXTRA_PARAM* Other dialogue command line arguments. The EXTRA_PARAM parameter allows sending other arguments by adding them to the DIALOGUE engine command line. The value of EXTRA_PARAM is simply added at the end of the DIALOGUE command line. EXTRA_PARAM can also point to the name of a stringlist object. At this time all object items are added to the command line as parameters.

*TIME_OUT* Time out in seconds. Maximum time for the dialogue engine to return. If not given or set to a negative value, the command waits indefinitely for dialogue to return (default).

*CTRL_FILE* Control file. When this parameter is set to yes, all command line parameters are written in a file given as control file to Dialogue. The default is “NO”.

### Objects created

*Output files* The COMPOSE command creates output file objects in the output container. The object names are defined by the output file maps of the document. For example, if the output file map has been defined as OUT:OUTPDF(PDF);OUT2:OUTAFP(AFP), the command will generate two file objects named “OUT” and “OUT2”. The attached files for these file objects have an extension corresponding to their defined format (“.pdf” and “.afp” in this example).

*OBJECTS* List of created file objects. It is a Nirva table object containing the following columns:

- “NAME” file object name.
- “FILENAME” associated file name.
- “EXTENSION” file extension.
- “SIZE” file object size.

## COMPOSE_DIRECT

DIALOGUE:DOCUMENT:COMPOSE_DIRECT

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | Yes | Yes |

### Description

This command is similar to the COMPOSE command but doesn’t use the workspace, package and document repositories of the DIALOGUE service. All the composition parameters must then be given in the command.

The COMPOSE_DIRECT command creates output file objects in the output container. The object names are defined by the output file maps of the document. For example, if the output file map has been defined as OUT:OUTPDF(PDF);OUT2:OUTAFP(AFP), the command will generate two file objects named “OUT” and “OUT2”. The attached file for these file objects have an extension corresponding to their defined format.

### Parameters

ENGINE

Engine version. This parameter can be given to specify a DIALOGUE engine version to use. If this parameter is not provided, NIRVA tries to evaluate the last engine version by alphabetically comparing the engine versions.

PFILE

Package file. This is the name of a NIRVA file object containing the DIALOGUE package file. The object must exist and point to the package file. The default value of PFILE is “PFILE”.

SUBTYPE

Sub-type can be used when a DIALOGUE package contains the definition of several documents. The subtype is a string transmitted to the DIALOGUE engine by a specific file map named “DOCTYPE”. At composition time, the DIALOGUE service creates the XML file containing the subtype and sends it to the engine as the “DOCTYPE” file map. The XML file has a single element named “DOCTYPE” that contains the subtype string. Here is an example of such a file:

```
<DOCTYPE>Mydoctype</DOCTYPE>
```

Where Mydoctype is the string given in the document subtype.

FORMAT

Default output format for the document. The output format can be “PDF”, “AFP”, “HTML”, “RTF”, “TIFF”, “PCL”, “XML”, “TXT”, “METACODE”, “POSTSCRIPT”, “VPS”, “IJPDS”, “VIPP”, “PPML” or “3211”. See the FILEMAPOUT parameter for information about output formats.

FILEMAPIN

Input file maps. This parameter is a collection of strings separated by a semicolon character and giving the mapping between NIRVA file objects and DIALOGUE input file maps. Each string has the following format: NirvaObject:DialogueFileMap where NirvaObject is the name of the NIRVA file object containing the input file and DialogueFileMap is the name of the DIALOGUE file map for this input. If the NirvaObject is given alone, the DIALOGUE file map is supposed to have the same name. The “DOCTYPE” file map name is reserved for transmitting the subtype to DIALOGUE and cannot be used.

FILEMAPOUT

Output file maps. This parameter is a collection of strings separated by a semicolon character and giving the mapping between NIRVA file objects and DIALOGUE output file maps. Each string has the following format: NirvaObject:DialogueFileMap(OutputFormat) where NirvaObject is the name of the NIRVA file object containing the output file, DialogueFileMap is the name of the DIALOGUE file map for this output and OutputFormat is the output file format. If the NirvaObject is given alone, the DIALOGUE file map is supposed to have the same name. If the OutputFormat is omitted, NIRVA uses the default output format (FORMAT parameter) defined for this document. The output formats can be “PDF”, “AFP”, “HTML”, “RTF”, “TIFF”, “PCL”, “XML”, “TXT”, “METACODE”, “POSTSCRIPT”, “VPS”, “IJPDS”, “VIPP”, “PPML” or “3211”. The “DOCTYPE” file map name is reserved for transmitting the subtype to DIALOGUE and cannot be used.

File maps
All the input file maps defined in the document parameters must be given as NIRVA file objects in the input container. For example, if the file map IN:INPUT has been defined, the IN file object must exist in the input container.

LOG
Keeps DIALOGUE log information file or not. When this parameter is set to “YES”, NIRVA creates an output file object named “MESSAGE” in the output container that gives log information about the DIALOGUE process.

CREATE
Create mode. If one of the output file objects already exists, NIRVA doesn’t create it again but changes the content of its associated file. One can force the object creation by setting the CREATE parameter to “YES”. The default is “NO”.

DIRECTORY
By default, NIRVA creates the output file objects in the application work directory. This can be changed by setting this parameter to the name of a directory.

EXTENSION
The created object file extensions are automatically set by the command following the required output formats. This can be overridden by giving a specific extension using this parameter. In this case, if there is more than one output file, all the output files will have the given extension. If the point character is omitted in the EXTENSION parameter, NIRVA adds it.

PREFIX
The PREFIX parameter is used in the file name when NIRVA creates the file objects.

SUFFIX
The SUFFIX parameter is used in the file name when NIRVA creates the file objects.

ERROR_NOSIZE
By default, the command doesn’t produce any error if one of the defined output file objects is empty. This can be changed by setting the ERROR_NOSIZE parameter to “YES”. In this case, if any of the output files is empty, the command returns an error.

EXTRA_PARAM
Other dialogue command line arguments. The EXTRA_PARAM parameter allows sending other arguments by adding them to the dialogue engine command line. The value of EXTRA_PARAM is simply added at the end of the dialogue command line.

CTRL_FILE
Control file. When this parameter is set to yes, all command line parameters are written in a file given as control file to Dialogue. The default is “NO”.

### Objects created

Output files

The COMPOSE_DIRECT command creates output file objects in the output container. The object names are defined by the output file maps of the document. For example, if the output file map has been defined as OUT:OUTPDF(PDF);OUT2:OUTAFP(AFP), the command will generate two file objects named “OUT” and “OUT2”. The attached files for these file objects have an extension corresponding to their defined format (“.pdf” and “.afp” in this example).

OBJECTS

List of created file objects. It is a Nirva table object containing the following columns:

- “NAME” file object name.
- “FILENAME” associated file name.
- “EXTENSION” file extension.
- “SIZE” file object size.

## CREATE

DIALOGUE:DOCUMENT:CREATE

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | No |

### Description

This command creates a new document entry in the document repository.

### Permissions

ADMIN

### Parameters

*WORKSPACE*

Workspace name. The workspace must exist.

*DOCUMENT*

Document name. This parameter uniquely identifies a document. The document name is mandatory. It is case independent and can contain spaces.
If a document with the same name already exists, the command fails.

*DESCRIPTION*

Document description. This is a free text that describes the document. Not mandatory.

*PACKAGE*

Name of the package that processes this document. It must be an existing package defined in the package repository of the same workspace. Several package names can be defined (multi packages feature). They must then be separated by a semicolon character (;).

*SUBTYPE*

The subtype can be used when a DIALOGUE package contains the definition of several documents. The subtype is a string transmitted to the DIALOGUE engine by a specific file map named “DOCTYPE”. At composition time, the DIALOGUE service creates the xml file containing the subtype and sends it to the engine as the “DOCTYPE” file map. The XML file has a single element named “DOCTYPE” that contains the subtype string. Here is an example of such a file:

```
<DOCTYPE>Mydoctype</DOCTYPE>
```

Where *Mydoctype* is the string given in the document subtype.

*FORMAT*

Default output format for the document. The output format can be “PDF”, “AFP”, “HTML”, “RTF”, “TIFF”, “PCL”, “XML”, “TXT”, “METACODE”, “POSTSCRIPT”, “VPS”, “IJPDS”, “VIPP”, “PPML” or “3211”. Please refer to the FILEMAPOUT parameter for information about output formats.

*FILEMAPIN*

Input file maps. This parameter is a collection of strings separated by the semicolon character and giving the mapping between NIRVA file objects and DIALOGUE input file maps. Each string has the following format: *NirvaObject*:*DialogueFileMap* where *NirvaObject* is the name of the NIRVA file object containing the input file and *DialogueFileMap* is the name of the DIALOGUE file map for the input. If the NirvaObject is given alone, the DIALOGUE file map is supposed to have the same name. The “DOCTYPE” file map name is reserved for transmitting the subtype to DIALOGUE and cannot be used.

If *DialogueFileMap* is set to “NV_DYNAMIC_PACKAGE”, Nirva considers the input file as a dynamic package and sends the file to DIALOGUE as a package instead of as a file map.

*FILEMAPOUT*

Output file maps. This parameter is a collection of strings separated by the semicolon character and giving the mapping between NIRVA file objects and DIALOGUE output file maps. Each string has the following format: *NirvaObject*:*DialogueFileMap*(*OutputFormat*) where *NirvaObject* is the name of the NIRVA file object containing the output file, *DialogueFileMap* is the name of the DIALOGUE file map for the output and *OutputFormat* is the output file format. If the NirvaObject is given alone, the DIALOGUE file map is supposed to have the same name. If the OutputFormat is omitted, NIRVA uses the default output format (FORMAT parameter) defined for this document. The output formats can be “PDF”, “AFP”, “HTML”, “RTF”, “TIFF”, “PCL”, “XML”, “TXT”, “METACODE”, “POSTSCRIPT”, “VPS”, “IJPDS”, “VIPP”, “PPML” or “3211”. The “DOCTYPE” file map name is reserved for transmitting the subtype to DIALOGUE and cannot be used.

| | |
|---|---|
| *WAIT* | Waiting time for the workspace to be free: in seconds. To create a new document entry, the DIALOGUE service must wait for the workspace to be free. This parameter allows defining the maximum waiting time. If the workspace is still in use after the waiting time, the command fails.<br>A value of 0 means: wait indefinitely.<br>The default is “10”. |
| *EXTRA_PARAM* | Document level extra parameters added to the Dialogue command line. The separator between parameters is line feed (\n). |

## INFO

### DIALOGUE:DOCUMENT:INFO

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | Yes |

#### Description

This command returns information about a specific document.

#### Parameters

| | |
|---|---|
| *WORKSPACE* | Workspace name. |
| *DOCUMENT* | Document name. |

#### Objects created

*DOCUMENT _INFO* It is a Nirva indexed string list object containing following keys:

- “NAME” is the document name.
- “DESCRIPTION” is the document description.
- “PACKAGE” is the name of the associated package
- “SUBTYPE” is the document subtype (DIALOGUE type)
- “FORMAT” is the default output format. It can be blank or of the following values: “PDF”, “AFP”, “HTML”, “RTF”, “TIFF”, “PCL”, “XML”, “TXT”, “METACODE”, “POSTSCRIPT”, “VPS”, “IJPDS”, “VIPP”, “PPML” or “3211”.
- “FILEMAPIN” is the input file maps. It is a collection of strings separated by the semicolon character and giving the mapping

between NIRVA file objects and DIALOGUE input file maps. Each string has the following format: *NirvaObject*:*DialogueFileMap* where *NirvaObject* is the name of the NIRVA file object containing the input file and *DialogueFileMap* is the name of the DIALOGUE file map for the input.

- “FILEMAPOUT” is the output file maps. It is a collection of strings separated by a semicolon character and giving the mapping between NIRVA file objects and DIALOGUE input file maps. Each string has the following format: *NirvaObject*:*DialogueFileMap*(*OutputFormat*) where *NirvaObject* is the name of the NIRVA file object containing the output file, *DialogueFileMap* is the name of the DIALOGUE file map for this output and *OutputFormat* is the output file format. The output file format can be blank or of the following values: “PDF”, “AFP”, “HTML”, “RTF”, “TIFF”, “PCL”, “XML”, “TXT”, “METACODE”, “POSTSCRIPT”, “VPS”, “IJPDS”, “VIPP”, “PPML” or “3211”. If the output file format is blank, the parentheses are omitted.
- “ENGINE” is the engine version eventually associated to the document.
- “EXTRA_PARAM contains document level extra parameters added to the dialogue command line (separator is line feed \n).

## LIST

DIALOGUE:DOCUMENT:LIST

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | Yes |

### Description

This command returns the list of the DIALOGUE service documents.

The command creates a table object returning the following document information:

- Name.
- Description.
- Associated package
- Subtype

- Default output format
- Input file map
- Output file map.
- Extra parameters.

### Parameters

| | |
|---|---|
| WORKSPACE | Workspace name. |

### Objects created

DOCUMENT_LIST

It is a Nirva table object containing following columns:

- “NAME” is the document name.
- “DESCRIPTION” is the document description.
- “PACKAGE” is the name of the associated package
- “SUBTYPE” is the document subtype (DIALOGUE type)
- “FORMAT” is the default output format. It can be blank or of one of the following values: “PDF”, “AFP”, “HTML”, “RTF”, “TIFF”, “PCL”, “XML”, “TXT”, “METACODE”, “POSTSCRIPT”, “VPS”, “IJPDS”, “VIPP”, “PPML” or “3211”.
- “FILEMAPIN” is the input file maps. It is a collection of strings separated by the semicolon character and giving the mapping between NIRVA file objects and DIALOGUE input file maps. Each string has the following format: NirvaObject:DialogueFileMap where NirvaObject is the name of the NIRVA file object containing the input file and DialogueFileMap is the name of the DIALOGUE file map for the input.
- “FILEMAPOUT” is the output file maps. It is a collection of strings separated by a semicolon character and giving the mapping between NIRVA file objects and DIALOGUE input file maps. Each string has the following format: NirvaObject:DialogueFileMap(OutputFormat) where NirvaObject is the name of the NIRVA file object containing the output file, DialogueFileMap is the name of the DIALOGUE file map for the output and OutputFormat is the output file format. The output file format can be blank or of one of the following values: “PDF”, “AFP”, “HTML”, “RTF”, “TIFF”, “PCL”, “XML”, “TXT”, “METACODE”, “POSTSCRIPT”, “VPS”, “IJPDS”, “VIPP”, “PPML” or “3211”. If the output file format is blank, the parentheses are omitted.
- “EXTRA_PARAM contains document level extra parameters added to the dialogue command line.

## REMOVE

DIALOGUE:DOCUMENT:REMOVE

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | No |

### Description

This command removes an existing document from the DIALOGUE documents list. This command may fail if the workspace is in use (see the WAIT parameter).

All information maintained by a document is removed by this command.

### Permissions

ADMIN

### Parameters

| | |
|---|---|
| *WORKSPACE* | Workspace name. |
| *DOCUMENT* | Document name. |
| *WAIT* | Waiting time for the workspace to be free: in seconds. To remove a document, the DIALOGUE service must wait for its workspace to be free. This parameter allows defining the maximum waiting time. If the workspace is still in use after the waiting time, the command fails.<br>A value of 0 means: wait indefinitely.<br>The default is “10”. |

## SET_PARAM

DIALOGUE:DOCUMENT:SET_PARAM

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | No |

### Description

This command changes the document parameters. The document must exist.

Any parameter can be changed except the document name. If a parameter is not provided, its value is unchanged.

### Permissions

ADMIN

### Parameters

WORKSPACE

Workspace name. The workspace must exist.

DOCUMENT

Document name. The document must exist.

DESCRIPTION

Document description. It is a free text that describes the document. Not mandatory.

PACKAGE

Name of the package that processes the document. It must be an existing package defined in the package repository of the same workspace. Several package names can be defined (multi packages feature). They must then be separated by a semicolon character (;).

SUBTYPE

Sub-type can be used when a DIALOGUE package contains the definition of several documents. The subtype is a string transmitted to the DIALOGUE engine by a specific file map named “DOCTYPE”. At composition time, the DIALOGUE service creates the XML file containing the subtype and sends it to the engine as the “DOCTYPE” file map. The XML file has a single element named “DOCTYPE” that contains the subtype string. Here is an example of such a file:

```
<DOCTYPE>Mydoctype</DOCTYPE>
```

where Mydoctype is the string given in the document subtype.

FORMAT

Default output format for the document. The output format can be “PDF”, “AFP”, “HTML”, “RTF”, “TIFF”, “PCL”, “XML”, “TXT”, “METACODE”, “POSTSCRIPT”, “VPS”, “IJPDS”, “VIPP”, “PPML” or “3211”. See the FILEMAPOUT parameter for information about output formats.

FILEMAPIN

Input file maps. This parameter is a collection of strings separated by a semicolon character and giving the mapping between NIRVA file objects and DIALOGUE input file maps. Each string has the following format: NirvaObject:DialogueFileMap where NirvaObject is the name of the NIRVA file object containing the input file and DialogueFileMap is the name of the DIALOGUE file map for this input. If the NirvaObject is given alone, the DIALOGUE file map is supposed to have the same name. The “DOCTYPE” file map name is reserved for transmitting the subtype to DIALOGUE and cannot be used.

| | |
|---|---|
| FILEMAPOUT | Output file maps. This parameter is a collection of strings separated by a semicolon character and giving the mapping between NIRVA file objects and DIALOGUE output file maps. Each string has the following format: NirvaObject:DialogueFileMap(OutputFormat) where NirvaObject is the name of the NIRVA file object containing the output file, DialogueFileMap is the name of the DIALOGUE file map for the output and OutputFormat is the output file format. If the NirvaObject is given alone, the DIALOGUE file map is supposed to have the same name. If the OutputFormat is omitted, NIRVA uses the default output format (FORMAT parameter) defined for the document. The output formats can be “PDF”, “AFP”, “HTML”, “RTF”, “TIFF”, “PCL”, “XML”, “TXT”, “METACODE”, “POSTSCRIPT”, “VPS”, “IJPDS”, “VIPP”, “PPML” or “3211”. The “DOCTYPE” file map name is reserved for transmitting the subtype to DIALOGUE and cannot be used. |
| WAIT | Waiting time in seconds for the workspace to be free. To set the document parameters, the DIALOGUE service must wait for its workspace to be free. This parameter allows defining the maximum waiting time. If the workspace is still in use after the waiting time, the command fails.<br>A value of 0 means: wait indefinitely.<br>The default is “10”. |
| EXTRA_PARAM | Document level extra parameters added to the Dialogue command line. The separator between parameters is line feed (\n). |

## ENGINE class

The ENGINE class provides commands for managing DIALOGUE engine versions. One can create, remove, change or list engine versions.

The DIALOGUE engine is an executable that does the composition step following information provided in one or several input files and a in configuration file named package.

The service doesn’t provide the DIALOGUE engines; they must be installed somewhere on the target machine. The NIRVA DIALOGUE service only requires the pathnames of the engine executables.

### CREATE

DIALOGUE:ENGINE:CREATE

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | No |

### Description

This command creates a new engine entry in the DIALOGUE service engine list.

### Permissions

ADMIN

### Parameters

*VERSION*

Engine version. This parameter uniquely identifies an engine version. The engine version is mandatory. It is case independent. The version number used by the NIRVA DIALOGUE service is independent from the real DIALOGUE version number. It is though advised to choose a string containing only numeric digits and dots, always with the same number of digits because NIRVA can automatically choose the last engine version when running a composition. This is done by alphabetically sorting the version numbers.
The VERSION parameter is mandatory. If an entry with the same version number already exists, the command fails.

*PATH*

Engine executable path. The PATH parameter must point to the DIALOGUE engine command line. One can set some extra parameters to the executable. Generally, it is not necessary because the complete command line is built at composition time by the NIRVA service.

*KEY*

License key of your DIALOGUE engine. This parameter is not mandatory since the key can be embedded directly into the engine itself. If your DIALOGUE reseller sends you a license key file, the real key can be extracted from this file by editing it.

## INFO

DIALOGUE:ENGINE:INFO

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | Yes |

### Description

This command returns information about a specific engine entry.

### Parameters

| | |
|---|---|
| *VERSION* | Engine version. |

### Objects created

*ENGINE_INFO*

This is a Nirva indexed string list object containing following keys:

- “VERSION” is the engine version number.
- “PATH” is the engine executable path.
- “KEY” is the engine license key.

## LIST

DIALOGUE:ENGINE:LIST

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | Yes |

### Description

This command returns the list of DIALOGUE service engine versions.

The command creates a table object returning the following engine information:

- Version.
- Path.
- Key.

### Parameters

None.

### Objects created

*ENGINE_LIST*

It is a Nirva table object containing the following columns:

- “VERSION” is the engine version number.
- “PATH” is the engine executable path.
- “KEY” is the engine license key.

## REMOVE

DIALOGUE:ENGINE:REMOVE

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | No |

### Description

This command removes an existing engine entry from the DIALOGUE service engine list. This command may fail if the engine is in use (see the WAIT parameter).

### Permissions

ADMIN

### Parameters

| | |
|---|---|
| VERSION | Engine version. |
| WAIT | Waiting time in seconds for the engine to be free. To remove an engine entry, the DIALOGUE service must wait for the engine to be free. This parameter allows defining the maximum waiting time. If the engine is still in use after the waiting time, the command fails.<br>A value of 0 means: wait indefinitely.<br>The default is “10”. |

## SET_PARAM

DIALOGUE:ENGINE:SET_PARAM

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | No |

### Description

This command changes the engine entry parameters. The engine entry must exist.

Any parameter can be changed except the engine version. If a parameter is not provided, its value is unchanged.

### Permissions

ADMIN

### Parameters

| | |
|---|---|
| VERSION | Engine version entry to set. |
| PATH | Engine executable path. The PATH parameter must point to the DIALOGUE engine command line. One can set some extra parameters to the executable. Generally, it is not necessary because the complete command line is built at composition time by the NIRVA service. |
| KEY | License key of your DIALOGUE engine. If your DIALOGUE reseller sends you a license key file, the real key can be extracted from this file by editing it. |
| WAIT | Waiting time in seconds for the engine to be free. To set the engine entry parameters, the DIALOGUE service must wait for the engine to be free. This parameter allows defining the maximum waiting time. If the engine is still in use after the waiting time, the command fails.<br>A value of 0 means: wait indefinitely.<br>The default is “10”. |

## PACKAGE class

The PACKAGE class provides commands for managing DIALOGUE packages. NIRVA maintains a repository of DIALOGUE packages for each workspace.

A DIALOGUE package is a binary file created with DIALOGUE tools that contains all the necessary configuration information for the DIALOGUE engine to run the composition for one or several document types.

The NIRVA DIALOGUE service allows hiding the packages to the final user, who is only interested with document types.

### CREATE

DIALOGUE:PACKAGE:CREATE

| Source | Use Input Container | Use Output Container |
|---|---|---|

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | No |

### Description

This command creates a new package entry in the package repository.

### Permissions

ADMIN

### Parameters

| | |
|---|---|
| *WORKSPACE* | Workspace name. The workspace must exist. |
| *PACKAGE* | Package name. This parameter uniquely identifies a package. The package name is mandatory. It is case independent and can contain spaces.<br>If a package with the same name already exists, the command fails. |
| *DESCRIPTION* | Package description. It is a free text that describes the package. Not mandatory. |
| *ENGINE* | Engine version to use for the package. This parameter, if provided, must correspond to an existing engine entry in the engines list. |
| *PFILE* | Package file. This is the name of a NIRVA file object containing the real DIALOGUE package file. If this object exists, NIRVA imports the file it points to as the package file.<br>The default value of PFILE is “PFILE”. |
| *WAIT* | Waiting time in seconds for the workspace to be free. To create a new package entry, the DIALOGUE service must wait for the workspace to be free. This parameter allows defining the maximum waiting time. If the workspace is still in use after the waiting time, the command fails.<br>A value of 0 means: wait indefinitely.<br>The default is “10”. |

## INFO

DIALOGUE:PACKAGE:INFO

| Source | Use Input Container | Use Output Container |
|---|---|---|

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | Yes |

### Description

This command returns information about a specific package.

### Parameters

| | |
|---|---|
| *WORKSPACE* | Workspace name. |
| *PACKAGE* | Package name. |

### Objects created

*PACKAGE _INFO* It is a Nirva indexed string list object containing following keys:

- “NAME” is the package name.
- “DESCRIPTION” is the package description.
- “ENGINE” is the package engine version.

## LIST

### DIALOGUE:PACKAGE:LIST

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | Yes |

### Description

This command returns the list of DIALOGUE service packages.

The command creates a table object returning the following package information:

- Name.
- Description.
- Engine version.

### Parameters

*WORKSPACE* Workspace name.

### Objects created

*PACKAGE_LIST* It is a Nirva table object containing following columns:

- “NAME” is the package name.
- “DESCRIPTION” is the package description.
- “ENGINE” is the package engine version.

## REMOVE

### DIALOGUE:PACKAGE:REMOVE

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | No |

### Description

This command removes an existing package from the DIALOGUE package list. This command may fail if the workspace is in use (see the WAIT parameter).

All information maintained by a package is removed by this command including package files.

### Permissions

ADMIN

### Parameters

*WORKSPACE* Workspace name.

*PACKAGE* Package name.

*WAIT* Waiting time in seconds for the workspace to be free. To remove a package, the DIALOGUE service must wait for its workspace to be free. This parameter allows defining the maximum waiting time. If the workspace is still in use after the waiting time, the command fails.
A value of 0 means: wait indefinitely.
The default is “10”.

## SET_PARAM

DIALOGUE:PACKAGE:SET_PARAM

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | Yes | No |

### Description

This command changes the package parameters. The package must exist.

Any parameter can be changed except the package name. If a parameter is not provided, its value is unchanged.

### Permissions

ADMIN

### Parameters

| | |
|---|---|
| *WORKSPACE* | Workspace name. The workspace must exist. |
| *PACKAGE* | Package name. The package must exist. |
| *DESCRIPTION* | Package description. It is a free text that describes the package. |
| *ENGINE* | Engine version to use for the package. This parameter, if provided, must correspond to an existing engine entry in the engine list. |
| *PFILE* | Package file. It is the name of a NIRVA file object containing the real DIALOGUE package file. If this object exists, NIRVA imports the file it points to as the package file. If the object doesn't exist, the actual package file is not changed.<br>The default value of PFILE is “PFILE”. |
| *WAIT* | Waiting time in seconds for the workspace to be free. To set the package parameters, the DIALOGUE service must wait for its workspace to be free. This parameter allows defining the maximum waiting time. If the workspace is still in use after the waiting time, the command fails.<br>A value of 0 means: wait indefinitely.<br>The default is “10”. |

### GET_FILE

DIALOGUE:PACKAGE:GET_FILE

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | Yes |

#### Description

This command retrieves the package file.

#### Parameters

WORKSPACE — Workspace name. The workspace must exist.

PACKAGE — Package name. The package must exist.

PFILE — Package file. It is the name of a NIRVA file object where the command must put the package file. If this object doesn't exist, NIRVA creates it.
The default value of PFILE is “PFILE”.

#### Objects created

PFILE — It is a Nirva file object containing the searched packaged file.

## WORKSPACE class

The WORKSPACE class provides commands for managing workspaces.

A workspace encapsulates the entire necessary production environment for a dedicated application. A workspace is a named object that includes the package and document type repositories.

### CREATE

DIALOGUE:WORKSPACE:CREATE

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | No |

### Description

This command creates a new workspace.

### Permissions

ADMIN

### Parameters

WORKSPACE

Workspace name. This parameter uniquely identifies a workspace. The workspace name is mandatory. It is case independent and can contain spaces.
If a workspace with the same name already exists, the command fails.

DESCRIPTION

Workspace description. This is a free text that describes the workspace. Not mandatory.

DEF_ENGINE

Default engine version to use for the workspace. This parameter, if provided, must correspond to an existing engine entry in the engines list. Generally, the engine version to use is defined at package level but NIRVA can use the value given in this parameter when no engine is defined at package level.

## EXPORT

DIALOGUE:WORKSPACE:EXPORT

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | Yes |

### Description

The export command saves all the workspace information into a single file including document and package repositories. It can be used in conjunction with the IMPORT command for deployment purposes.

### Permissions

ADMIN

### Parameters

*WORKSPACE*

Workspace name.

*EXPORT* — Name of the NIRVA file object that should contain the exported workspace. The default is “EXPORT”. The command creates the file object in the output container.

### Objects created

*EXPORT* — NIRVA file object containing the exported workspace information. The name of the object can be changed using the EXPORT parameter.

## IMPORT

DIALOGUE:WORKSPACE:IMPORT

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | Yes | No |

### Description

The import command creates or changes the entire content of a workspace with the information given in an import file. The import file must be the result of a previous EXPORT command.

The workspace name is included into the workspace but can be changed by a dedicated parameter in order to create a copy of a workspace.

The workspace is created if it doesn’t previously exist.

### Permissions

ADMIN

### Parameters

*WORKSPACE* — Workspace name. Not mandatory. If the workspace name is not provided, the command takes it directly from the import file.

*IMPORT* — Name of the NIRVA file object containing the import file. The default is “IMPORT”.

*WAIT* — Waiting time for the workspace to be free: in seconds. This parameter has meaning only if the imported workspace already exists. At this time, the DIALOGUE service must wait for the workspace to be free. This parameter allows defining the maximum waiting time. If the workspace is still in use after the waiting time, the command fails.

A value of 0 means: wait indefinitely.
The default is “10”.

## INFO

### DIALOGUE:WORKSPACE:INFO

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | Yes |

#### Description

This command returns information about a specific workspace.

#### Parameters

| | |
|---|---|
| WORKSPACE | Workspace name. |

#### Objects created

WORKSPACE_INFO

It is a Nirva indexed string list object containing following keys:

- “NAME” is the workspace name.
- “DESCRIPTION” is the workspace description.
- “DEF_ENGINE” is the workspace default engine version.

## LIST

### DIALOGUE:WORKSPACE:LIST

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | Yes |

### Description

This command returns the list of the DIALOGUE service workspaces.

It creates a table object returning the following workspace information:

- Name.
- Description.
- Default engine version.

### Parameters

None.

### Objects created

*WORKSPACE_LIST* It is a Nirva table object containing following columns:

- “NAME” is the workspace name.
- “DESCRIPTION” is the workspace description.
- “DEF_ENGINE” is the workspace default engine version.

## REMOVE

DIALOGUE:WORKSPACE:REMOVE

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | No |

### Description

This command removes an existing workspace from the DIALOGUE workspace list. This command may fail if the workspace is in use (see the WAIT parameter).

All information maintained by a workspace is removed by this command, document types and package repositories included.

### Permissions

ADMIN

### Parameters

WORKSPACE

Workspace name.

WAIT

Waiting time for the workspace to be free: in seconds. To remove a workspace, the DIALOGUE service must wait for the workspace to be free. This parameter allows defining the maximum waiting time. If the workspace is still in use after the waiting time, the command fails.
A value of 0 means: wait indefinitely.
The default is “10”.

## SET_PARAM

DIALOGUE:WORKSPACE:SET_PARAM

| Source | Use Input Container | Use Output Container |
|---|---|---|
| Client<br>Web<br>Procedure<br>Service | No | No |

### Description

This command changes the workspace parameters. The workspace must exist.

Any parameter can be changed except the workspace name. If no parameter is provided, its value is unchanged.

### Permissions

ADMIN

### Parameters

*WORKSPACE*

Workspace name.

*DESCRIPTION*

Workspace description. It is a free text that describes the workspace.

*DEF_ENGINE*

Default engine version to use for the workspace. This parameter, if provided, must correspond to an existing engine entry in the engine list. Generally, the engine version to use is defined at package level but NIRVA can use the value given in this parameter when no engine is defined at package level.

*WAIT*

Waiting time for the workspace to be free: in seconds. To set the workspace parameters, the DIALOGUE service must wait for the workspace to be free. This parameter allows defining the maximum waiting time. If the workspace is still in use after the waiting time, the command fails.

A value of 0 means: wait indefinitely.

The default is “10”.