nirva

Virtual printer Interface - Nirva

Document version: 1.17

Virtual printer version: 3.30

nirva

Table of Contents

nirva

Overview

This document describes the technical interface between the Nirva virtual printer connector and the Nirva application running on the Nirva server.

The interface is carried out in three steps:

- Connection to the directory to retrieve technical parameters.
- Transmission and processing of the file.
- Access to the service portal.

The term VP or Virtual Printer represents the virtual printing agent installed on the client machine.

Directory

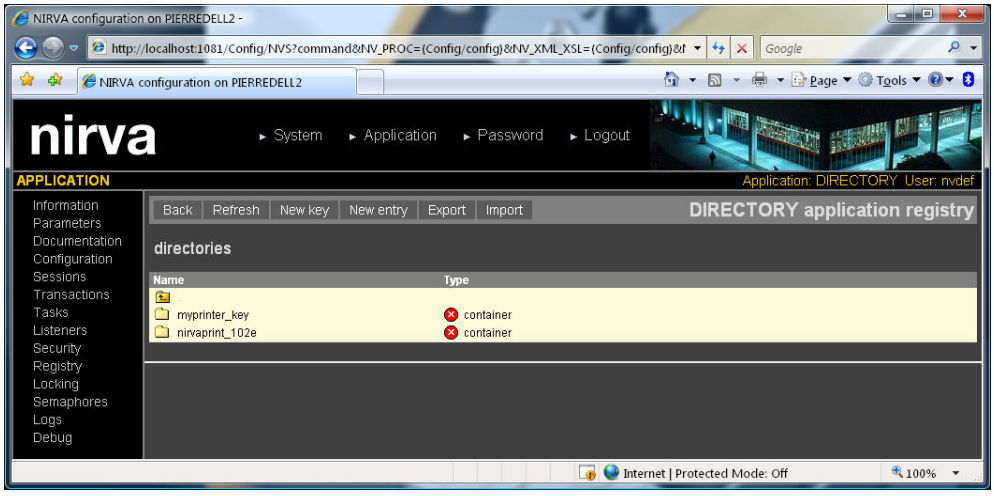
Directory description

The directory allows the central configuration of one or many virtual printers.

The directory is a Nirva application containing a key called ‘directories’ in its registry. The key itself contains a series of sub-keys corresponding to specified names called ‘directory keys’. The directory application must contain a procedure called GetData.nvp stored in its Procs directory.

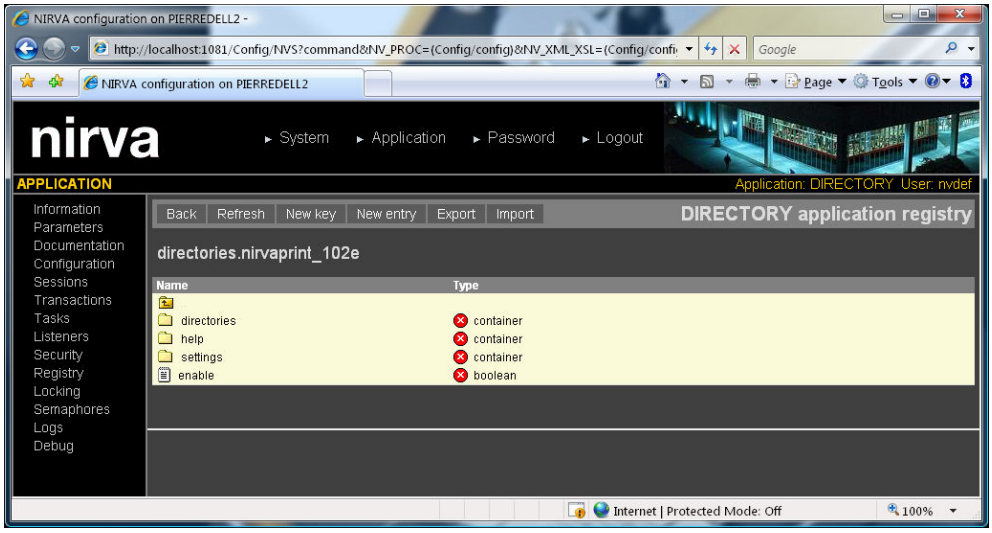
The nvdef user must be available and must not bear a password. It is recommended to define a specific Nirva application for the directory and to give no access rights to the nvdef user.

Directory keys



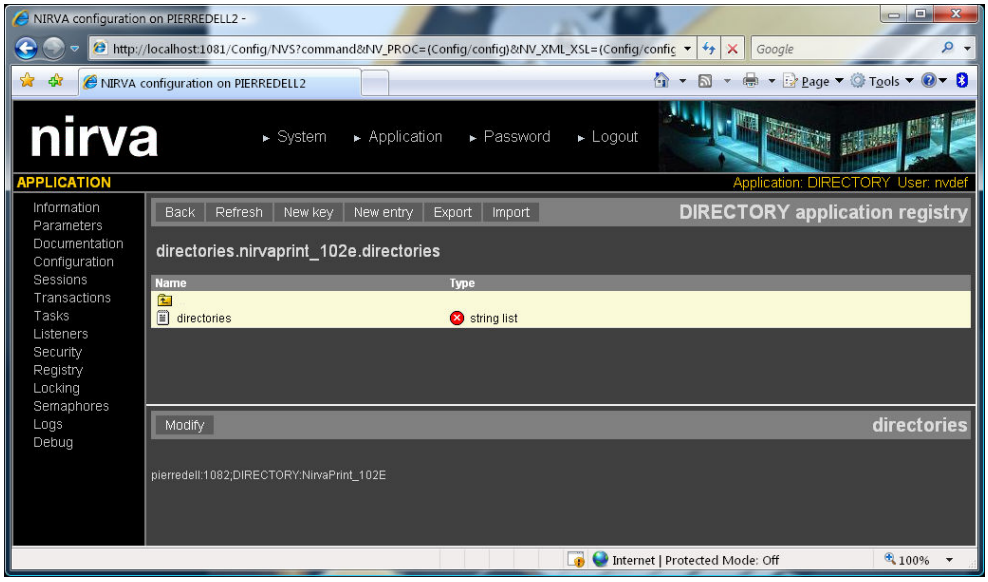
A directory key is composed of the following elements:

- A Nirva object of boolean type called ‘enable’ indicating whether the element is active or not.
- A ‘directories’ sub-key
- A ‘help’ sub-key
- A ‘settings’ sub-key



Directories

The ‘directories’ sub-key contains the list of directory addresses defined in an object of type string list also called ‘directories’.



These directory lists allow the connection of the virtual printer to the correct directory. When the VP is installed, a directory address list is configured by default and is automatically updated when connecting the directory using the ‘directories’ sub-key.

A directory address shows the following format:

`server:port;directory:key`

Where `server` is the address or name of the Nirva server supporting the directory, `port` is the HTTPS port used for the connection; `directory` is the name of the directory application and `key` is the name of the directory key.

Here is an example of a directory address:

```
192.168.1.5:1082;DIRECTORY:NirvaPrint_102E
```

`192.168.1.5:1082` is the Nirva server address (with its port). `DIRECTORY` is the name of the Nirva directory application and `NirvaPrint_102E` is the name of the directory key.

The transfer protocol is always HTTPS and is therefore not specified.

Help

The ‘help’ sub-key contains the list of help URLs for the various virtual printer interface windows defined in an objet of type string list called ‘topics’.

When a URL associated to a help button is configured for a particular window, the virtual printer opens the Internet browser with this URL.

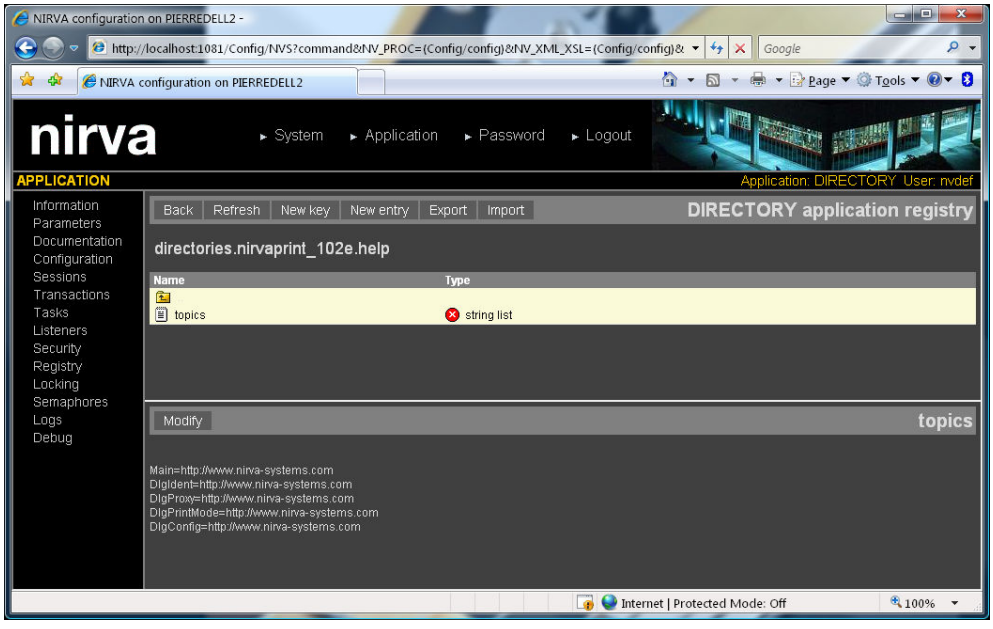
Each entry in the ‘topics’ object is a chain of characters in the form of:

`window_name=url`

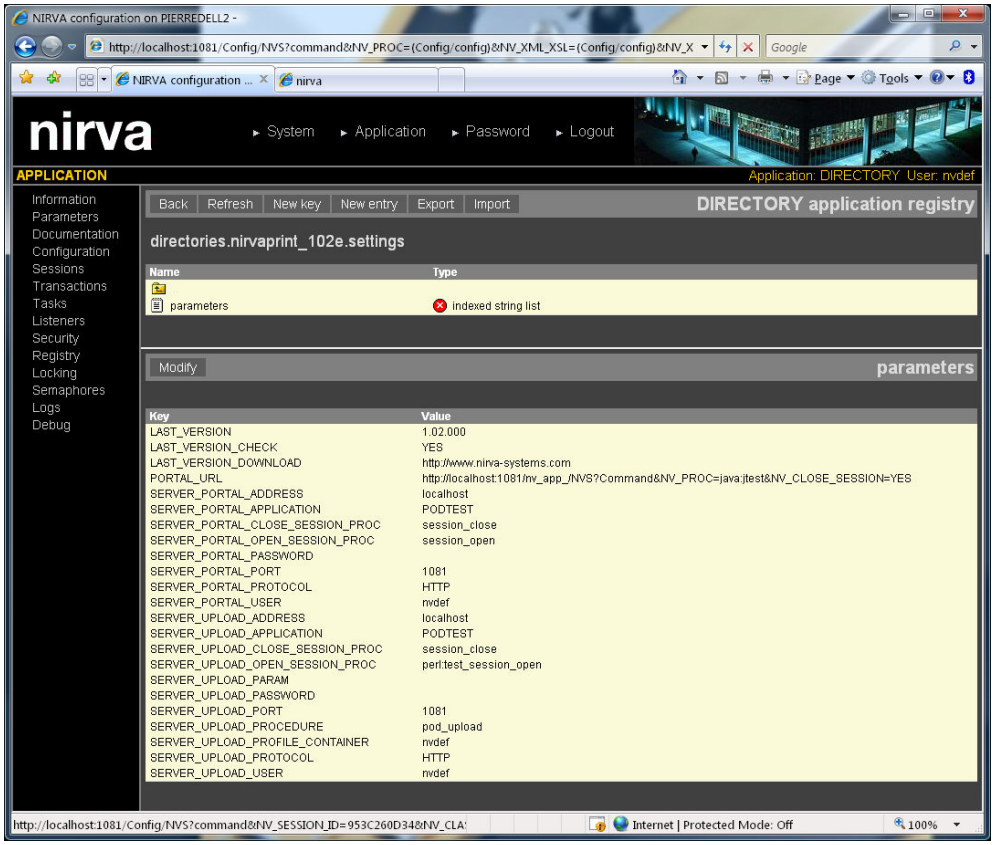
Where `window_name` is the name of the virtual printer window and `url` is the help URL associated with it.

window name is one of the following values:

- Main – Global help accessible via the ‘nvp –m help’ command of the virtual printer. A client side installer usually specifies an entry in the start menu pointing to this help topic.
- DlgIdent – Help associated with the user credentials window.
- DlgProxy – Help associated with the proxy parameters window.
- DlgPrintMode – Help associated with the print mode selection window.
- DlgConfig – Help associated with the configuration parameters window.
- DlgService – Help associated with the service choice window.



Settings



This key accesses the list of parameters of the virtual printer. It refers to an object of type indexed string list containing the following:

LAST_VERSION	Latest version of the virtual printer. If version control is activated and the virtual printer is not in its latest version, it will prompt the user about the availability of a newer version. The version number is in the form of `MM.mm.bbb` where `MM` is the major version number, `mm` is the minor version number and `bbb` is the build number.
LAST_VERSION_CHECK	Value of 'YES' or 'NO'. Activates or deactivates version control.
LAST_VERSION_DOWNLOAD	Download URL for the latest version of the virtual printer.
PORTAL_URL	URL called by the virtual printer when the print mode is 'standard'. The virtual printer automatically appends the NV_SESSION_ID parameter to this URL.
SERVER_PORTAL_ADDRESS	TCP/IP address or name of the portal server. If the portal is hosted on more than one server in load balancing mode behind a switch, the address is the switch's in question. The switch must be configured to synchronise itself on the session ID since the portal

	session is opened by the virtual printer and subsequently connected by the browser. The Nirva servers must be configured to use 'long sessions ID's'.
SERVER_PORTAL_APPLICATION	Name of the Nirva application managing the portal.
SERVER_PORTAL_CLOSE_SESSION_PROC	Name of the procedure closing the portal session.
SERVER_PORTAL_OPEN_SESSION_PROC	Name of the procedure opening the portal session.
SERVER_PORTAL_PASSWORD	Nirva password for the opening of the portal session (usually empty as user management is generally not handled by Nirva security).
SERVER_PORTAL_PORT	TCP/IP port for portal connection.
SERVER_PORTAL_PROTOCOL	HTTP or HTTPS. Protocol for portal connection.
SERVER_PORTAL_USER	Name of Nirva user for portal connection (usually nvdef).
SERVER_PORTAL_SSO	If set to "YES", enable SSO for the portal application when the authentication mode for this application is 'Nirva or Single Sign-On'. If the application is not defined to use SSO or if it is defined to always use SSO, this parameter is not used. The default is "NO".
SERVER_PORTAL_SSO_PRINCIPAL	Service Principal Name (SPN) when using SSO with Kerberos (not required for NTLM). This must correspond to a SPN set on the domain controller. See Nirva documentation for further information about SSO.
SERVER_UPLOAD_ADDRESS	TCP/IP address or name of the server hosting the print file retrieval application (upload). There could be more than one address separated by a semi-colon (';'). In this case, the virtual printer selects one of the addresses at random. This allows for simple load balancing.
SERVER_UPLOAD_APPLICATION	Name of the Nirva application managing the retrieval of the print file (upload).
SERVER_UPLOAD_CLOSE_SESSION_PROC	Name of the procedure closing the upload session.
SERVER_UPLOAD_OPEN_SESSION_PROC	Name of the procedure opening the upload session.
SERVER_UPLOAD_PARAM	Optional parameters to be sent to the procedure managing the print file. These parameters must adhere to the standard Nirva command syntax.
SERVER_UPLOAD_PASSWORD	Nirva password for the upload session (usually empty as user management is generally not handled by Nirva security).
SERVER_UPLOAD_PORT	TCP/IP port for connection to the upload server.
SERVER_UPLOAD_PROCEDURE	Name of the procedure to launch to process the uploaded file.

SERVER_UPLOAD_PROFILE_CONTAINER	Name of the session container that contains user profile information.
SERVER_UPLOAD _PROTOCOL	HTTP or HTTPS. Protocol for the connection to the upload server.
SERVER_UPLOAD_USER	Name of the Nirva user for the connection to the upload server (usually nvdef).
SERVER_UPLOAD_SSO	If set to “YES”, enable SSO for the upload application when the authentication mode for this application is ‘Nirva or Single Sign-On’. If the application is not defined to use SSO or if it is defined to always use SSO, this parameter is not used. The default is “NO”.
SERVER_UPLOAD_SSO_PRINCIPAL	Service Principal Name (SPN) when using SSO with Kerberos (not required for NTLM). This must correspond to a SPN set on the domain controller. See Nirva documentation for further information about SSO.
ENABLE_FILTER	Control the way the filtering of the input file is authorized. If set to “NEVER” (default value, the filter is never authorized. If set to “ALWAYS” the filtering is authorized (if one is defined in the virtual printer configuration). If set to “PROFILE”, the authorization is controlled by a parameter in the user’s profile “PARAMETERS” object. This object is an indstringlist object, the key is “FILTER” and the value is “YES” (default) or “NO”.
SERVER_UPLOAD_BLOCK_SIZE	Size of the block of data used when sending a file to the server in bytes. The minimum value is 1024. The default value is 65536.
SERVER_UPLOAD_NUM_ATTEMPTS	Number of attempts to ask user name and password if identification failed. The default value is 1.
USE_EXTERNAL_AUTH	Enable the use of an external program to check security. The external program name must be configured in the config.txt file. Can take values “NO” or “YES”. The default value is “NO”.
EXTERNAL_AUTH_PARAM	Optional parameters that will be added to the external security program called by the virtual printer when the USE_EXTERNAL_AUTH param has been set to “YES”.

Printing

When a user is printing a document using the virtual printer, the following operations take place:

- Connection to the directory to retrieve the virtual printer parameters
- Sending of the print file in the format specified by the printer (usually Postscript) to the upload server
- Optional launch of the portal to adjust service parameters associated to the file management

Directory connection

In order to connect the directory, the VP must first obtain its address. For that purpose, it first tries to connect the last successfully accessed directory. This information is stored during a previous connection in the windows registry (HKEY_LOCAL_MACHINE/Software/*Company*/*PrinterSoftware*/Settings section where *Company* is the company name defined in the configuration file and *PrinterSoftware* is the software name defined in the configuration file). If not successful, the VP gets the list of available directories from the windows registry at machine level (this list is itself updated from the directory after a successful connection to a directory or directly set from the configuration file after an installation). If not successful, the VP gets the list of available directories from the windows registry at user level (this list is itself updated from the directory after a successful connection to a directory).

Once the connection is established, the VP triggers a procedure called ‘GetData’ using the name of the directory key to connect as a parameter (parameter DIRECTORY)

Other procedure parameters include:

- KEY (sub-key of the directory key to retrieve).
- ENTRIES (name of the objects to retrieve).
- SUBKEYS (optional retrieval of sub-keys)

In fact, these three last parameters correspond to the parameters of the same name used in the Nirva command REGISTRY:GET (please refer to Nirva documentation).

The GetData procedure retrieves registry information with the requested parameters and stores retrieved objects in the default session container (nvdef).

The GetData procedure is called several times with the following parameters and corresponding retrieved objects:

- Retrieval of the ‘Settings’ key
  - Parameters
    - DIRECTORY=|directory_name| where directory_name is the name of the directory key,
    - KEY=|Settings|
    - ENTRIES=|Parameters|
  - Retrieved objects
    - Parameters.
- Retrieval of the ‘Directories’ key
  - Parameters
    - DIRECTORY=|directory_name| where directory_name is the name of the directory key,
    - KEY=|Directories|
    - ENTRIES=|Directories|
  - Retrieved objects
    - Directories.
- Retrieval of the ‘Help’ key
  - Parameters
    - DIRECTORY=|directory_name| where directory_name is the name of the directory key,
    - KEY=|Help|
    - ENTRIES=|Topics|
  - Retrieved objects
    - Topics.

The implementation of the GetData.nvp procedure is under the responsibility of the Directory application. Here is an implementation example:

```
GetData.nvp:

; Retrieve data from directory and put it in the output container
; Just call the perl procedure that does the real job
NV_CMD=|COMMAND:GET_PARAMETER| NAME=|DIRECTORY| NV_VAR=|DIR_NAME|
NV_CMD=|COMMAND:GET_PARAMETER| NAME=|KEY| NV_VAR=|DIR_KEY|
NV_CMD=|COMMAND:GET_PARAMETER| NAME=|ENTRIES| NV_VAR=|DIR_ENTRIES|
NV_CMD=|COMMAND:GET_PARAMETER| NAME=|SUBKEYS| NV_VAR=|DIR_SUBKEYS|
NV_PROC=|perl:getdata| DIRECTORY=|#DIR_NAME| KEY=|#DIR_KEY| ENTRIES=|#DIR_ENTRIES|
SUBKEYS=|#DIR_SUBKEYS|

GetData.pl:

NV::GetParameter("DIRECTORY");
$DIRECTORY=CleanString($NV::RESULT);
if(!length($DIRECTORY))
```

```
{
  NV::SetError("No directory name");
  exit();
}

NV::GetParameter("KEY");
$KEY=CleanString($NV::RESULT);
NV::GetParameter("ENTRIES");
$ENTRIES=CleanString($NV::RESULT);
NV::GetParameter("SUBKEYS");
$SUBKEYS=CleanString($NV::RESULT);

# Check if the directory key exists or not
NV::Command("NV_CMD=|REGISTRY:EXIST| NV_CONTAINER=|DIR_TMP_CONTAINER|
KEY=|directories.$DIRECTORY|");
NV::Command("NV_CMD=|OBJECT:BOOLEAN_GET_VALUE| NV_CONTAINER=|DIR_TMP_CONTAINER|
NAME=|EXIST|");
if($NV::RESULT ne "TRUE")
{
  NV::Command("NV_CMD=|CONTAINER:REMOVE| NAME=|DIR_TMP_CONTAINER|");
  NV::SetError("The directory doesn't exist");
  exit();
}

# Check if enabled or not (default is yes)
NV::Command("NV_CMD=|REGISTRY:GET| NV_CONTAINER=|DIR_TMP_CONTAINER|
KEY=|directories.$DIRECTORY| ENTRIES=|enable| NV_NO_ERROR=|YES|");
NV::Command("NV_CMD=|OBJECT:BOOLEAN_GET_VALUE| NV_CONTAINER=|DIR_TMP_CONTAINER|
NAME=|ENABLE| NV_NO_ERROR=|YES|");
if($NV::RESULT eq "FALSE")
{
  NV::Command("NV_CMD=|CONTAINER:REMOVE| NAME=|DIR_TMP_CONTAINER|");
  NV::SetError("The directory is disabled");
  exit();
}
NV::Command("NV_CMD=|CONTAINER:REMOVE| NAME=|DIR_TMP_CONTAINER|");

# So the directory exists and is enabled
# We can get the required data into the output container
# If no key and entries required, we do nothing. This is just a check
# that the directory exists and is enabled
if((!length($KEY)) && (!length($ENTRIES)))
{
  exit();
}
$REALKEY=directories . "." . $DIRECTORY;
if(length($KEY))
{
  $REALKEY=$REALKEY . "." . $KEY;
}
NV::Command("NV_CMD=|REGISTRY:GET| KEY=|$REALKEY| ENTRIES=|$ENTRIES| SUBKEYS=|$SUBKEYS|
APPEND=|YES| REPLACE=|YES|");

sub CleanString
{
  my $STRING = @_[0];
  $STRING =~ tr/\t/ /;
  $STRING =~ tr/ //s;
```

```
  $STRING =~ tr/\n//;
  $STRING =~ tr/\r//;
  $STRING =~ s/^\s+//;
  $STRING =~ s/\s+$//;
  return $STRING;
}
```

When the directory parameters have been retrieved, the virtual printer closes the session and the connection to the directory.

Sending the file

The virtual printer opens a connection to the upload server with the connection parameters retrieved from the directory.

Before the connection, the VP prompts the user for connection credentials if these have not been saved and if token connection mode is not used or failed (see later). Credentials include user name and password. User name is a chain containing two values: account and username, separated by colon ':'. If the user name is not supplied the administrator user is assumed (admin). For example, if 'st1:usr1' is user 'usr1' of account 'st1'. 'myemail@myprovider.com' is the 'admin' user of account 'myemail@myprovider.com'. The user ID and password are transmitted during the first order sent to the upload server via the 'POD_USER' and 'POD_PASSWORD' parameters.

If SSO has been activated, these parameters are not transmitted. The name of the SSO user is then available server side in the NV_SSO_USER Nirva variable. The account name must therefore be hard coded on the server. In order to activate SSO the upload application must be configured in SSO mode to identify users (please refer to Nirva documentation) and the SSO directory parameters must be configured.

If external security authentication has been activated, the virtual printer calls an external program defined in the configuration file. This external program must return a token that will be send to the server in base64 in the POD_EXT_TOKEN parameter. It is the responsibility of the server to check this token and to allow or not the user. If the external security program cannot authenticate the user it must return a value different of 0 (so 0 in case of success). If the error code is 1, the virtual printer displays a message "authentication failed" and stops. If the error code is 2, the virtual printer stops without displaying any message. External security have priority on SSO.

On server side, the open session procedure (as defined in the directory) is responsible for checking the security options and to retrieve the user profile. The user profile must be supplied in a container whose name is declared in the directory (SERVER_UPLOAD_PROFILE_CONTAINER parameter). The user profile may contain an indstringlist object named "PARAMETERS" that contains extra parameters for the user.

When checking user and password (or POD_EXT_TOKEN when using the external security), the procedure may produce nirva errors SYSTEM:SECURITY:101 (bad user) and SYSTEM:SECURITY:103 (bad password). The virtual printer checks these error codes and displays the corresponding error message (BadUser and BadPassword entries in the Errors section of the config file).

The server may implement the token mode. Token mode allows connecting users without requesting their password if they successfully connected previously. The token mode doesn't work when SSO or external

security are used. In order to use the token mode, the application must provide a token inside a string object in the user's profile container (in the session open procedure). The virtual printer then stores this token in the local user's settings and sends it back to the server into a parameter named POD_TOKEN at the next connection. The server can then use the token to verify the user and may accept or refuse the connection. In order to refuse the connection it must produce the SYSTEM:SECURITY:103 (bad password) error. Then the virtual printer retries the connection by sending the user's password (asks the user's password if this one wasn't saved). This is the responsibility of the application to manage and store the user's tokens.

The virtual printer retrieves an object called 'PRINTING_MODES' from the profile container. It is a Nirva object of stringlist type containing allowed print modes for this user. These modes include 'STANDARD' for standard mode (call to services portal), 'TUNNEL' for tunnel mode (automated processing), 'ASYNC' for the asynchronous mode (differed processing) and dynamic (processing controlled by the server).

If multiple modes are available, the user is prompted for a choice.

If the mode is 'TUNNEL', the profile container may contain a string object named TUNNEL_MODE that can be set to "SINGLE" or "MULTIPLE". If this object does not exist, the default value is "SINGLE".
In single mode, the virtual printer retrieves from the profile container the name of the service that will process the file as well as the name of the service profile (service parameters). These are respectively stored in objects of type string called 'TUNNEL_SERVICE_NAME' and 'TUNNEL_SERVICE_PROFILE' located in the profile container.
In multiple mode, the virtual printer retrieves from the profile container an object of type table named TUNNEL_SERVICES that contains the service names, descriptions and profiles that can be used by the user. The table object has three columns named "NAME", "DESCRIPTION" and "PROFILE" containing respectively the service name, the service description and the service profile name. All columns are mandatory. If there is more than one value, the virtual printer prompts the user for his choice. The names displayed for this selection are the ones stored in the DESCRIPTION column.

The virtual printer filters the file to send to the server if a filter has been defined and authorized (see the Filter key in the directory section for information about filter authorization).

The virtual printer sends the file to be processed in the session container NVDEF on the upload server. The name of the object of type file is 'RECEIVED_FILE'. It is a non persistent object.

The virtual printer sends the metadata in the NVDEF container as an object of type indstringlist called 'FILE_INFO'. It contains the following entries:

PRINT_MODE	Print mode. Values are 'STANDARD', 'TUNNEL', 'ASYNC' or 'DYNAMIC'.
PRINTER_NAME	Name of the virtual printer used to send the file.
FILE_NAME	Name of the original local file name. This is the name of the document to be printed as received by the virtual printer from the application.
FILE_SIZE	Size of the file.
WIN_MACHINE	Name of the client Windows machine. May be empty.
WIN_USER	Windows user name. May be empty.
TUNNEL_SERVICE	Name of the service for tunnel mode processing. Empty if the mode is not 'TUNNEL'.
TUNNEL_PROFILE	Name of the service profile for tunnel mode processing. Empty if the mode is not 'TUNNEL'.

IV_VERSION	Virtual printer version as defined in the config file. This version is a parameter of the virtual printer and can be changed in the config file when installing the virtual printer.
IV_INTERNAL_VERSION	Virtual printer internal version. This is the official version of the software. It changes only when the source code of the virtual printer changes.

The virtual printer then asks the server to launch the procedure to process the file as defined in the directory (SERVER_UPLOAD_PROCEDURE parameter). The optional parameters defined in the directory are sent to the server. If the process fails (Nirva error), the virtual printer displays an error message and stops.

The virtual printer then retrieves the unique file identifier created by the server in an object of type string called 'FILE_ID' and stored in the NVDEF container.

If the printing mode has been set to 'DYNAMIC', the virtual printer tries to get 2 strings object respectively named "MESSAGE" and "LINK" from the NVDEF container. "MESSAGE" contains a message that is displayed by the virtual printer (ex: "Your file has been processed successfully"). The message may contain some basic html tags (see http://doc.qt.io/qt-4.8/richtext-html-subset.html#supported-css-selectors for the list of supported tags) . "LINK" contains an URL that the virtual printer calls in the user's default browser.

Finally, the virtual printer closes the session and the connection to the upload server.

Calling the portal

Access to the service portal is only triggered if the mode is 'STANDARD' and the file has been successfully received.

The virtual printer first opens a session on the portal server using the connection parameters as defined in the directory and the credentials keyed in by the user prior to file transmission. The user name and password are sent during the first order sent to the portal server within the 'POD_USER' and 'POD_PASSWORD' parameters. Server side, the session open procedure (as defined in the directory) checks security and retrieves the user profile.

If SSO has been activated for the portal, these parameters are not transmitted. The name of the SSO user is then available server side in the NV_SSO_USER Nirva variable. In order to activate SSO the portal application must be configured in SSO mode to identify users (please refer to Nirva documentation) and the SSO directory parameters must be configured.

The virtual printer then retrieves the session ID.

The virtual printer sends the file identifier in the form of an object of type string called 'FILE_ID' in the NVDEF container of the session.

The virtual printer then closes the connection (but not the session) and opens an internet browser pointing to the URL specified in the directory, adding the NV_SESSION_ID parameters (with the session ID opened previously) and CTRL. CTRL is a control key that allows verifying that the Web user is the same as the virtual printer user. The key is calculated in the following way:

```
UpperCase(MD5(UpperCase(SESSION_ID::POD_USER::POD_PASSWORD))).
```

The key must be verified by the Nirva procedure that manages the URL (this is not mandatory but strongly recommended). An error message must be displayed in the browser if the calculated key is different from the key passed as a parameter.