ESPF – Technical Specifications

Version A5

Table of contents

1. The ESPF concept

The ESPF is a server and the gateway to the EPS features (supervision mode) using requests. All communications leverage IPC standard protocols (Socket or Named Pipe).

The request process is based on the *JSON-RPC* protocol, which eases the enablement of clients as shown in the next sections.

All data presented in this document is based on version 1.7.0.157 of the ESPF server and version 3.0 of the ESPF request language.

2. Overview of the services

The EPS features available in the ESPF are categorized per service. For each service, such features can be accessed by calling a set of methods. Selected services require the implementation of a specific call sequence.

The ESPF provides the following services:

SERVICE NAME	PURPOSE
CMD	Bidirectional direct communication with the printer
PRINT	Direct printing of a card
SETTING	Printer configuration
SUPERVISION	Management of the printer’s status
ECHO	Test service
ADD-ON	Management of add-on execution
ESPF	Server configuration

3. Request model

The request model used between the clients and the ESPF service leverages JSON-RPC v2.0, a light-weight and stateless RPC (Remote Procedure Call) protocol using the *JSON* format

The *Notification* and *Batch* concepts of *JSON-RPC v2.0* are not supported for the ESPF. The protocol is used in compliance with the following specifications:

- The `params` member is an object which member names matching the parameters expected by the server. Any value is a *String*.
- The `id` member is a *String*. For a response to a processed request, but with an unknown *id*, the *id* value is considered as *Null*.
- The `result` member is a *String*
- The `data` member is not considered.

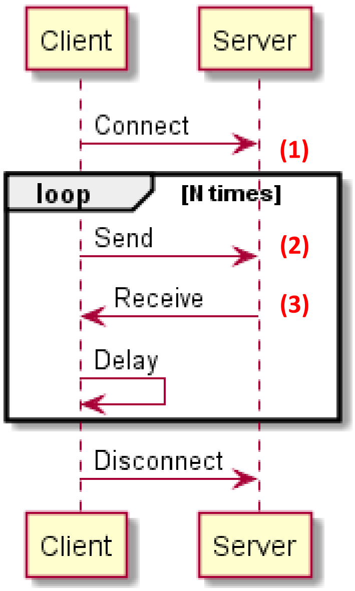
For calling a request, the client sends a *Request* object to the server and receives a reply from the server in the form of a *Response* objet.

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "CMD.SendCommand",
    "params": {
        "command": "Rfv",
        "timeout": "5000",
        "device": "Primacy1"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "1450"
}
```

This implementation, at the communication protocol level, must be carried out as shown below:



(1) During the connection, the IP address, the server port or the Named pipe are identified according to the IPC communication protocol used.
(2) The request is sent through a call on a full *buffer* (no sending when the *buffer* is sliced).
(3) The response to the request is expected to be a single reply.

Notes:

- If an unexpected error appears on the server while communicating with a connected client, the server will shut the communication channel with this client.
- If a client disconnects from the server, the server will stop any process in progress with this client.
- When sending a request, if the reception of the request takes longer that a configurable timeout (3,000 ms by default), the server will shut the communication channel with the client (support for « *half-open connections* » issues).

4. Services – Technical description

CMD Service

Purpose

- ➢ **Sends commands in text or binary format**
- ➢ **Receives statuses in binary format**
- ➢ **Reset communication**

SendCommand **METHOD**	**Sends commands in text or binary format**
command	text or binary commands
timeout	timeout in milliseconds
device	device name
result	response to the sent command
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "CMD.SendCommand",
    "params": {
        "command": "Rfv",
        "timeout": "5000",
        "device": "Primacy1"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "1450"
}
```

Note 1

For sending a binary command, data must be base64-encoded. The `command` parameter is made up of the encoded data complemented by the `base64:` prefix.

Here is an example of a base64-encoded `Rfv` command.

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "CMD.SendCommand",
    "params": {
        "device": "Primacy1",
        "command": "base64:G1Jmdg0=",
        "timeout": "3000"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "1450"
}
```

Note 2

If the response timeout is exceeded, the returned response will be an error with related error code.

If the `timeout` setting is not indicated, a default value will be provided. This value is of 30000 ms for the service. This parameter is not taken into account when the device is connected in Ethernet.

GetStatus **METHOD**	**Retrieves the binary status of a device**
device	device name
result	the binary status of the device and the current ID session (see appendix)
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "CMD.GetStatus",
    "params": {
        "device": "Primacy1"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "88D02420204000000000000000000000800C2000000000000000000000000000:32005"
}
```

Note

This method can be called even if the printer is printing a job or processing a command.

***ResetCom* METHOD**	**Reset communications with a device**
timeout	timeout in milliseconds
device	device name
result	`OK` if successful
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "CMD.ResetCom",
    "params": {
        "timeout": "10000",
        "device": "Primacy1"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "OK"
}
```

Note

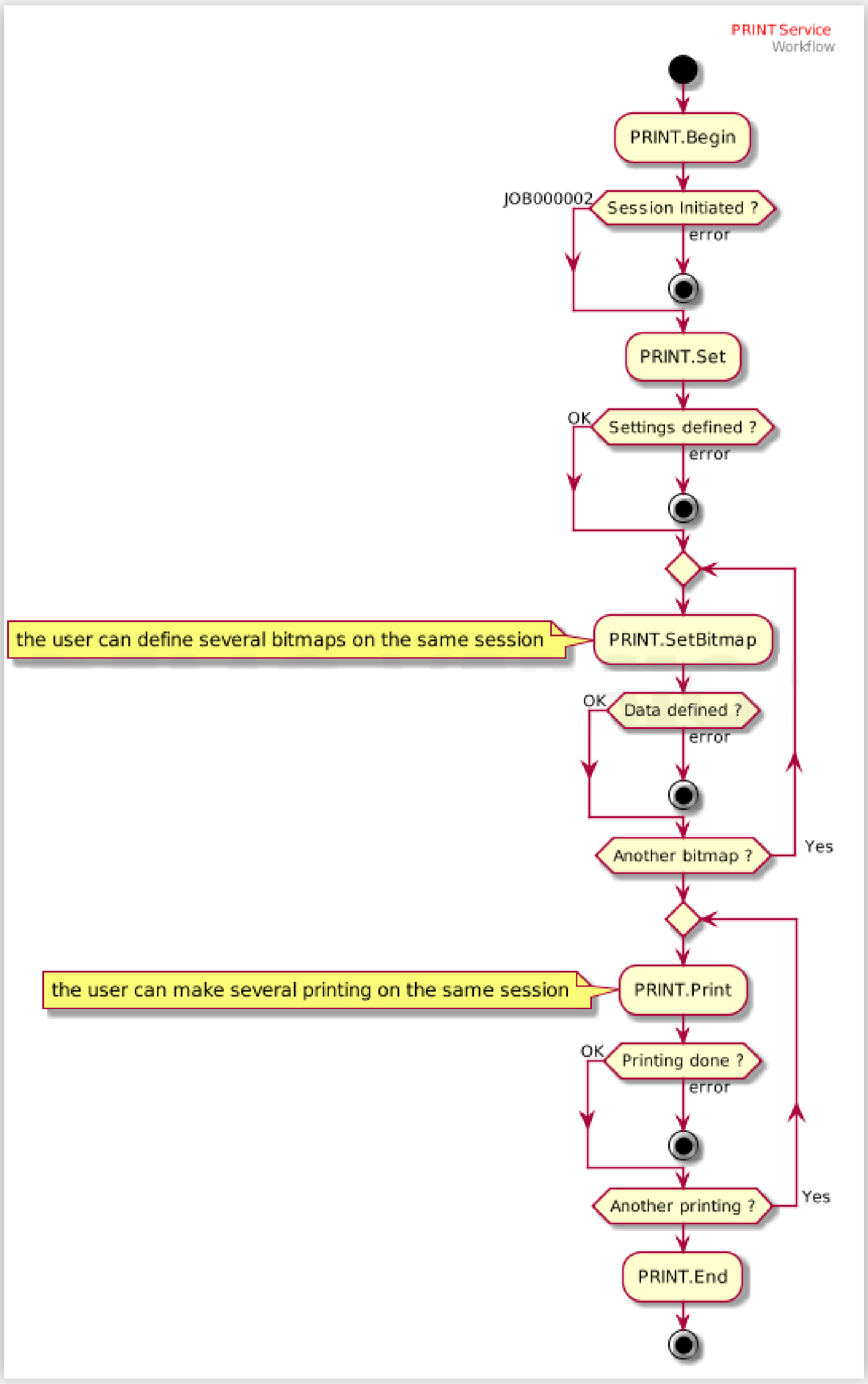
If the response timeout is exceeded, the returned response will be an error with related error code.

If the `timeout` setting is not indicated, a default value will be provided. This value is of 30000 ms for the service. This parameter is not taken into account when the device is connected in Ethernet.

PRINT Service

Purpose

- Prints a card, whatever the printing system
- Configures the printing jobs



***Begin* METHOD**	**Initiates a printing session**
device	device name
session	(optional: to set only if we don’t want to get a Job ID generated automatically by this function) Job ID to use for the printing session
result	Job ID if successful
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "PRINT.Begin",
    "params": {
        "device": "Primacy1"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "JOB000002"
}
```

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "PRINT.Begin",
    "params": {
        "device": "Primacy1",
        "session": "MYJOBID1"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": " MYJOBID1"
}
```

Note 1

Printing session: a timeframe during which a printer is assigned to one or several printing jobs specified in the following format if generated automatically: `JOBXXXXXX`.

Note 2

The job ID is not unique and is referenced in the printing logs.

Note 3

The service can manage only one printing session (job) at a time per printer instance.

`Set` METHOD	**Sets the printing parameters**
`session`	Job ID
`data`	printing parameters in the following format `key1=value1;…;keyN=valueN`
`result`	`OK` if successful
`error`	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "PRINT.Set",
    "params": {
        "session": "JOB000002",
        "data": "FColorBrightness=VAL12;GRibbonType=RC_YMCKO"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "OK"
}
```

Note

The `settings.xml` file describes all printing parameters and their possible values for each printer model. This file is available from:

%EPS_DIR%\Evolis Premium Suite\Model\{printer model}

`%EPS_DIR%` is either the installation directory for the suite or an environment variable.

To read the XML file, open it with an appropriate Web browser.

***SetBitmap* METHOD**	**Defines the graphic data to be printed**
session	Job ID
face	`front` or `back`
panel	`color`, `resin` or `varnish`
data	Image data (bitmap) encoded to base 64
result	`OK` if successful
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "PRINT.SetBitmap",
    "params": {
        "session": "JOB000002",
        "face": "front",
        "panel": "color",
        "data": "base64:Qk12Ix4AAAAAADYAAAAo...
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "OK"
}
```

Note 1

The method will be called only once per card side and panel type. The `color` panel type includes graphical data required for the three YMC panels.

Note 2

If the method is called twice or trice for a side, the `color` panel type will be the primary source, while the `black` and/or `overlay panel types` will be set as optional.

Print **METHOD**	**Launches a printing job**
session	Job ID
result	`OK` if successful
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "PRINT.Print",
    "params": {
        "session": "JOB000002"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "OK"
}
```

Note 1

The printing traffic is monitored during the printing job. This feature is always enabled and cannot be configured.

Note 2

When calling the `PRINT` method, the `GetEvent` method from the `SUPERVISION` service must be polled on a regular basis. If an event is identified, an action must be taken so that the print job can be finalized.

GetJobID **METHOD**	**Get current printing session Job ID**
device	device name
result	current Job ID if successful
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "PRINT.GetJobID",
    "params": {
        "device": "Primacy1"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "JOB000002"
}
```

***End* METHOD**	**Ends a printing session**	
session	Job ID	to set only if you want to close the printing session by Job ID
device	device name	to set only if you want to close the printing session by device name
result	`OK` if successful	
error	error code and related message (see appendix)	

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "PRINT.End",
    "params": {
        "session": "JOB000002"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "OK"
}
```

EXAMPLE

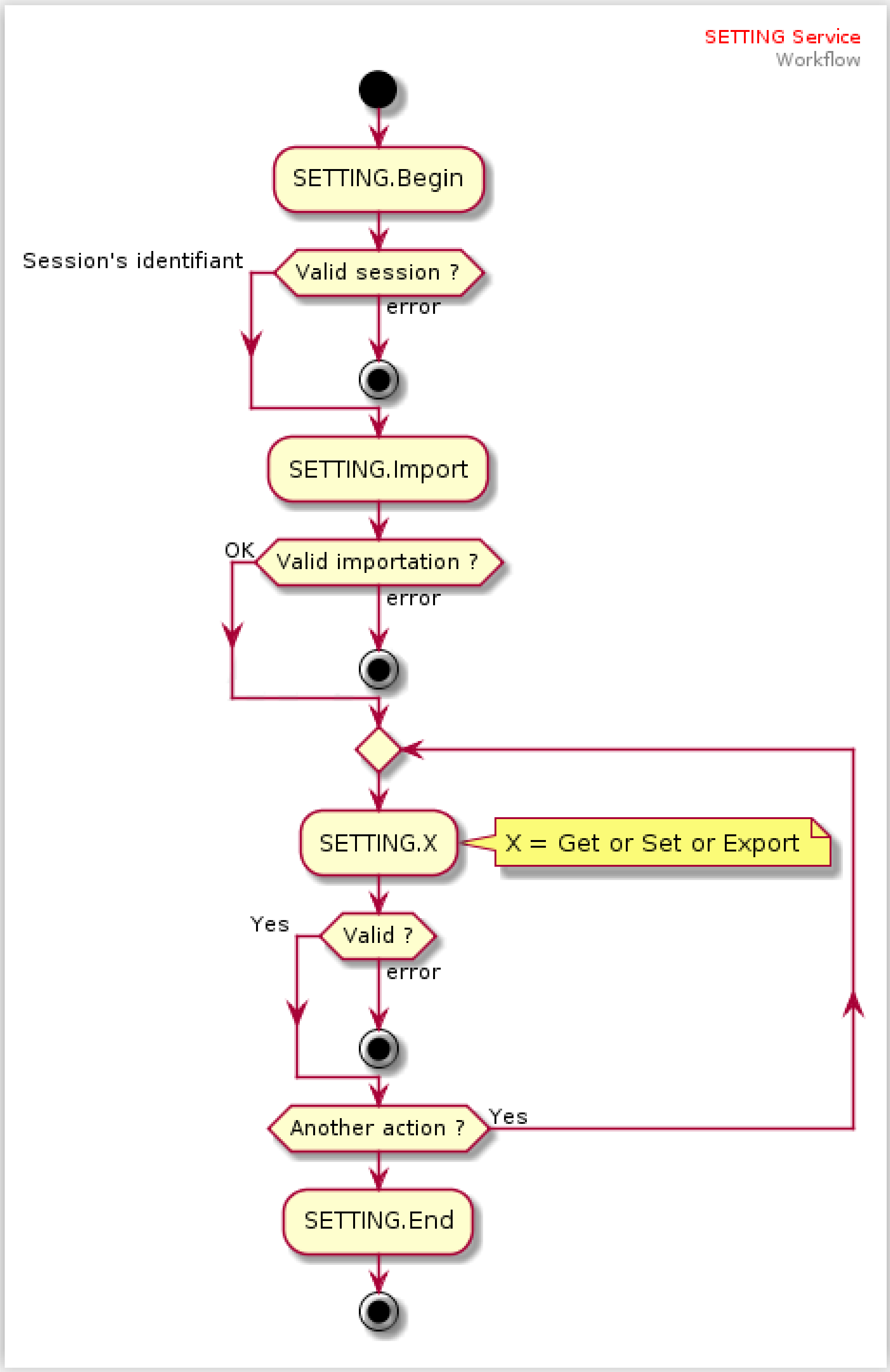
```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "PRINT.End",
    "params": {
        "device": "Primacy1"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "OK"
}
```

SETTING Service

Purpose

- **Loads printing parameters**
- **Reads the loaded parameters**
- **Edits the loaded parameters**
- **Exports the printing parameters**



***Begin* METHOD**	**Starts a configuration session**
device	device name
result	session ID
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "SETTING.Begin",
    "params": {
        "device": "Primacy1"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "SET000001"
}
```

***Import* METHOD**	**Imports parameters**	
session	session ID	
format	`printer`	existing parameters of the device
	`default`	default parameter for the printer model
	`xml`	from a parameter file (.dat)
data	base64-encoded data (for `xml` format)	
result	`OK` if successful	
error	error code and related message (see appendix)	

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "SETTING.Import",
    "params": {
        "session": "SET000001",
        "format": "printer"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "OK"
}
```

Note

With this method, parameters can be imported via the active session, and in a non-persistent way (until the end of the configuration session).

Get METHOD	**Gets the value of a parameter**
session	session ID
data	`key` parameter ID
result	`value` parameter value
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "SETTING.Get",
    "parameters": {
        "session": "SET000001",
        "data": "FColorContrast"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "15"
}
```

Set METHOD	**Edits the value of a parameter**
session	session ID
data	`key=value` ID of the parameter with its new value
result	`OK` if successful
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "SETTING.Set",
    "parameters": {
        "session": "SET000001",
        "data": "FColorContrast=18"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "OK"
}
```

***Export* METHOD**	**Exports parameters**	
session	Session ID	
format	`printer`	to the device
	`text`	in text format
	`xml`	to a data file (.dat) for the device

result		
	if `format=printer`	`OK` if successful
	if `format=text`	list of parameters in the following format : `key=value`
	if `format=xml`	export of base64-encoded configuration file
error	error code and related message (see appendix)	

EXAMPLE
```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "SETTING.Export",
    "params": {
        "session": "SET000001",
        "format": "text"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "BBlackManagement=ALLBLACKPOINT;
               BColorBrightness=VAL10;
               BColorContrast=VAL10;
               BHalftoning=THRESHOLD;
               BMonochromeContrast=VAL10;
               BOverlayContrast=VAL10;
               BOverlayManagement=FULLVARNISH;
               BPageRotate180=OFF;
               FBlackManagement=ALLBLACKPOINT;
               FColorBrightness=VAL10;
               FColorContrast=VAL13;
               FHalftoning=THRESHOLD;
               FMonochromeContrast=VAL10;
               FOverlayContrast=VAL10;
               FOverlayManagement=FULLVARNISH;
}
```

***End* METHOD**	**Ends the session**
session	session ID
result	`OK` if successful
error	error code and related message (see appendix)

EXAMPLE
```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "SETTING.End",
    "params": {
        "session": "SET000001"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "OK"
}
```

SUPERVISION Service

PURPOSE

- **Lists the subscribed devices and their state**
- **Subscribe/unsubscribe a device to a service, with state notification**

***List* METHOD**	**List all subscribed devices**
level	0 list of stateless devices 1 List of devices with major state (see appendix) 2 List of devices with a major or a minor state (see appendix)
device	name of the printer model type
result	`printername` (if `level=0`) `printername,majorstate` (if `level=1`) `printername,majorstate,minorstate` (if `level=2`)
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "SUPERVISION.List",
    "params": {
        "level": "2",
        "device": "Evolis Primacy"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "Primacy,READY,PRINTER_READY"
}
```

Note 1

If several printers are subscribed, the response to the request will have the following format, respectively for `level=2`, `level=1` and `level=0`:

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "Primacy,READY,PRINTER_READY;Primacy2,WARNING,FEEDER_EMPTY"
}
```

EXAMPLE
```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "Primacy,READY;Primacy2,WARNING"
}
```

EXAMPLE
```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "Primacy;Primacy2"
}
```

Note 2

- If the `level` parameter is not mentioned, then `level=0`
- The major states provide global information on the device.
- The minor states, offers more detailed information of the major states.
- The states reported by the `SUPERVISION` service are exactly the same states that are reported by the Print Center, before, after and during a print job (see appendix). If it is needed to test a specific status, for example not reported outside a print job by the Print Center, it is recommended to use the method `GetStatus` of the `CMD` service.

***AddDevice* METHOD**	**Subscribes a new device to the notification service**
device	device name
result	`OK` if successful
error	error code and related message (see appendix)

EXAMPLE
```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "SUPERVISION.AddDevice",
    "params": {
        "device": "Zenius1"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "OK"
}
```

***RemoveDevice* METHOD**	**Unsubscribes a device to the notification service**
device	device name
result	`OK` if successful
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "SUPERVISION.RemoveDevice",
    "params": {
        "device": "Zenius1"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "OK"
}
```

***GetState* METHOD**	**Requests the state of a device**
device	device name
result	`majorstate, minorstate` (see appendix)
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "SUPERVISION.GetState",
    "params": {
        "device": "Primacy1"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "READY,PRINTER_READY"
}
```

Note

The states reported by the `SUPERVISION` service are exactly the same states that are reported by the Print Center before, after and during a print job (see appendix). If it is needed to test a specific status, for example not reported outside a print job by the Print Center, it is recommended to use the method `GetStatus` of the `CMD` service.

***GetEvent* METHOD**	**Returns the notification of an unexpected event, as well as the list of actions for a device**
device	device name
result	`NONE` If no notification `minorstate:combination of actions` in case of a notification (see appendix)
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "SUPERVISION.GetEvent",
    "params": {
        "device": "Primacy1"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "ERR_MECHANICAL:CANCEL,RETRY"
}
```

Note

An unexpected event is notified during printing by the `PRINT` service or by the spooler. This process is related to the pop-up notifications from the Printer Manager.

***SetEvent* METHOD**	**Executes an action when an unexpected event is notified on a device during printing**
action	`minorstate:action` possible actions : `CANCEL`, `OK`, `RETRY`
device	device name
result	`OK` if successful
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "SUPERVISION.SetEvent",
    "params": {
        "action": "FEEDER_EMPTY:CANCEL",
        "device": "Primacy1"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "OK"
}
```

Note

Sending an action when notified of an unexpected event is equivalent to acting on the Printer Manager’s feedback notification. In this example, the feeder is empty and a request for cancelling the printing job is sent.

ECHO Service

Purpose

- **Checks that the *ESPF* is enabled**

Echo **METHOD**	**Sends a character string to the server**
data	A character string
result	the same string
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "ECHO.Echo",
    "params": {
        "data": "Hello World"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "Hello World"
}
```

ADD-ON Service

Purpose

- **Executes an add-on**

Launch **METHOD**	**Executes an application on the server**
command	name of the application
data	parameters of the application
result	return code of the application
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "ADDON.Launch",
    "params": {
        "command": "testplugin.exe"
        "data": "testdata1 testdata2 testdata3"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "0"
}
```

Note 1

- The `Launch` method triggers a response as soon as the application is fully executed.
- The application must be executed without involving the user.

Note 2

The `ADD-ON` service is disabled by default.

Note 3

The folder hosting the add-ons can be configured in the ESPF service (see appendix).

ESPF Service

Purpose

- Reads the server parameters
- Edits the server parameters

GetParam METHOD	Reads the value of a server parameter
key	parameter name
result	parameter value
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "ESPF.GetParam",
    "params": {
        "key": "ESPFServerManager.port"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "18000"
}
```

SetParam METHOD	Edits the value of server parameter
key	parameter name
data	new value to be set
result	`OK` if successful
error	error code and related message (see appendix)

EXAMPLE

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "method": "ESPF.SetParam",
    "params": {
        "key": "ESPFServerManager.port",
        "data": "18001"
    }
}
```

```
{
    "id": "1",
    "jsonrpc": "2.0",
    "result": "OK"
}
```

Note

The following parameters can be read:

- `ESPFService.version`: ESPF service version.
- `ESPFService.requestlanguageversion`: ESPF service request language version.

The following parameters can be read/set:

- `ESPFServerManager.port`: port number to be used by clients for socket-based communications with the server.
- `ESPFTcpServerConnectionSupervisor.receivetimeout`: timeout in ms before disconnection if no data is received via a socket-based communication with a client.
- `ESPFServerManager.tcpenabled`: the `true` value triggers the server's Socket layer, the `false` value disables it.

Other parameters can be edited only through the server's configuration file (see appendix).

The `Socket` layer parameters cannot be edited if the layer is enabled.

5. Appendix

5.1. Binary status of a printer

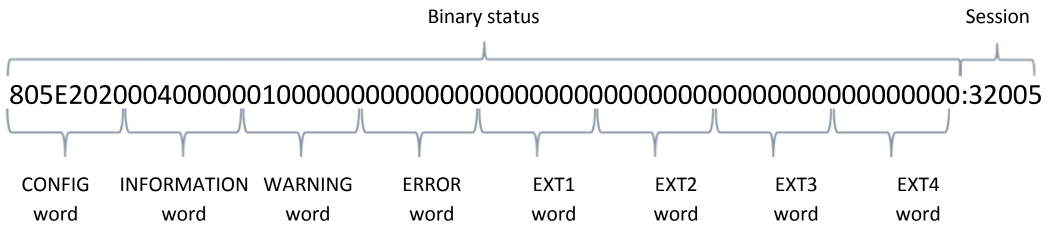
The binary status of a printer is a sequence of binary data giving the state of a printer at a given time. The printer should be set in `Pps;1 mode` (state is enabled in the printer's response frame, communication with the printer leverages a bidirectional protocol), so that the service identifies the state and the session ID.

The status offers different types of information:

- Printer hardware configuration (*CONFIG*),
- Printer state (*INFORMATION*),
- Warning messages (*WARNING*),
- Error message (*ERROR*),
- Extended data 1 (*EXT1*),
- Extended data 2 (*EXT2*),
- Extended data 3 (*EXT3*),
- Extended data 4 (*EXT4*).

The information is in the form of a 32 bytes data piece (8x4 bytes), with 4 bytes for each type of information listed above and according to the list order (32 bits of information possible for each type of information).

The state string returned by the service must be converted into a Hexadecimal value, so that it can be matched to the hexadecimal masks shown in the following table. Here is an explanation for each string section.



The following tables explain the bits meaning for each information type. The mask indicated for each bit is the hexadecimal mask to be used for testing the bit within the context of the word (32 bits) containing this bit, and in relation with one type of information.

Word containing *CONFIG type information*

Byte	Bit designation	Mask	Description
1	CFG_X01	0x80000000	Primacy printer model
	CFG_X02	0x40000000	Zenius printer model
	CFG_X04	0x10000000	Elypso printer model
	CFG_EXTENSION_1	0x08000000	Status Extension 1 enabled
2	CFG_WIFI	0x00800000	Wi-Fi is available
	CFG_ETHERNET	0x00400000	Ethernet is available
	CFG_FLIP	0x00100000	Flip-over feature is available
	CFG_CONTACTLESS	0x00080000	Contactless encoding feature is available
	CFG_SMART	0x00040000	Smart encoder is available
	CFG_MAGNETIC	0x00020000	Magnetic encoder is available
3			
	CFG_EXTENDED_RESOLUTION	0x00000400	See appendix
	CFG_LCD	0x00000200	LCD feature is available
4			
	CFG_JIS_MAG_HEAD	0x00000040	JIS magnetic encoding head is available
	CFG_MONO_ONLY	0x00000008	Monochrome printing only
	CFG_KC100	0x00000004	KC100 printer Model

Word containing *INFORMATION type information*

Byte	Bit designation	Mask	Description
1			
	INF_CARD_FEEDER	0x20000000	A card is present in the feeder module
	INF_CARD_FLIP	0x10000000	A card is present in the flip-over module
	INF_CARD_CONTACTLESS	0x08000000	A card is present in the Contactless module
	INF_CARD_SMART	0x04000000	A card is present in the Smart module
	INF_CARD_PRINT	0x02000000	A card is present in the printing module
	INF_CARD_EJECT	0x01000000	A card is present in the eject module
2			
	INF_SLEEP_MODE	0x00200000	The printer is in standby mode
	INF_UNKNOWN_RIBBON	0x00100000	See appendix
	INF_LOW_RIBBON	0x00080000	See appendix
	INF_CLEANING_MANDATORY	0x00040000	The cleaning cycle is exceeded
	INF_CLEANING	0x00020000	See appendix
3	INF_CLEAN_OUTWARRANTY	0x00008000	See appendix
	INF_CLEAN_LAST_OUTWARRANTY	0x00004000	See appendix
	INF_CLEAN_2ND_PASS	0x00002000	See appendix
	INF_CLEANING_ADVANCED	0x00000800	See appendix
	INF_WRONG_ZONE_RIBBON	0x00000400	See appendix
	INF_CLEANING_REQUIRED	0x00000100	See appendix
4	INF_PRINTING_RUNNING	0x00000080	See appendix
	INF_ENCODING_RUNNING	0x00000040	See appendix
	INF_CLEANING_RUNNING	0x00000020	See appendix
	INF_WRONG_ZONE_ALERT	0x00000010	See appendix
	INF_WRONG_ZONE_EXPIRED	0x00000008	See appendix
	INF_UPDATING_FIRMWARE	0x00000002	See appendix

Word containing *WARNING type information*

Byte	Bit designation	Mask	Description
1			
	DEF_CARD_ON_EJECT	0x04000000	See appendix
	DEF_WAIT_CARD	0x02000000	See appendix
	DEF_FEEDER_EMPTY	0x01000000	See appendix
2			
	DEF_COOLING	0x00200000	See appendix
	DEF_HOPPER_FULL	0x00100000	See appendix
	DEF_RIBBON_ENDED	0x00080000	See appendix
	DEF_PRINTER_LOCKED	0x00040000	See appendix
	DEF_COVER_OPEN	0x00020000	See appendix
	DEF_NO_RIBBON	0x00010000	See appendix
3	DEF_UNSUPPORTED_RIBBON	0x00008000	See appendix
4			

Word containing *ERROR type information*

Byte	Bit designation	Mask	Description
1			
	ERR_HEAD_TEMP	0x20000000	See appendix
	ERR_FEEDER_ERROR	0x08000000	See appendix
	ERR_RIBBON_ERROR	0x04000000	See appendix
	ERR_COVER_OPEN	0x02000000	See appendix
	ERR_MECHANICAL	0x01000000	See appendix
2	ERR_REJECT_BOX_FULL	0x00800000	See appendix
	ERR_BAD_RIBBON	0x00400000	See appendix
	ERR_RIBBON_ENDED	0x00200000	See appendix
			See appendix
	ERR_BLANK_TRACK	0x00080000	See appendix
	ERR_MAGNETIC_DATA	0x00040000	See appendix
	ERR_READ_MAGNETIC	0x00020000	See appendix
	ERR_WRITE_MAGNETIC	0x00010000	See appendix
3	ERR_FEATURE	0x00008000	See appendix
4			

Word containing *EXT1 type information*

Byte	Bit designation	Mask	Description
1	CFG_EXTENSION_2	0x80000000	Status Extension 2 enabled
2			
	CFG_LAMINATOR	0x00080000	Laminator feature is available
	INF_LAMINATING_RUNNING	0x00020000	See appendix
3	INF_LAMI_TEMP_NOT_READY	0x00008000	See appendix
	INF_FEEDER_NEAR_EMPTY	0x00000100	See appendix
4	INF_FEEDER1_EMPTY	0x00000080	Card feed problem Please check cards, position in the card feeder 1 and gauge adjustment.
	INF_FEEDER2_EMPTY	0x00000040	Card feed problem Please check cards, position in the card feeder 2 and gauge adjustment.
	INF_FEEDER3_EMPTY	0x00000020	Card feed problem Please check cards, position in the card feeder 3 and gauge adjustment.
	INF_FEEDER4_EMPTY	0x00000010	Card feed problem Please check cards, position in the card feeder 4 and gauge adjustment.
	INF_FEEDER1_NEAR_EMPTY	0x00000008	Feeder 1 almost empty The card feeder 1 is almost empty, please refill.
	INF_FEEDER2_NEAR_EMPTY	0x00000004	Feeder 2 almost empty The card feeder 2 is almost empty, please refill.
	INF_FEEDER3_NEAR_EMPTY	0x00000002	Feeder 3 almost empty The card feeder 3 is almost empty, please refill.
	INF_FEEDER4_NEAR_EMPTY	0x00000001	Feeder 4 almost empty The card feeder 4 is almost empty, please refill.

Word containing *EXT2 type information*

Byte	Bit designation	Mask	Description
1	CFG_EXTENSION_3	0x80000000	Status Extension 3 enabled
2	CFG_LAMINATION_MODULE_2	0x00800000	Laminator 2 feature is available
	INF_LAMINATE_UNKNOWN	0x00400000	See appendix
	INF_LAMINATE_LOW	0x00200000	See appendix
	INF_LAMI_CLEANING_RUNNING	0x00080000	See appendix
	INF_LAMI_UPDATING_FIRMWARE	0x00040000	See appendix
3	DEF_NO_LAMINATE	0x00008000	See appendix
	DEF_LAMI_COVER_OPEN	0x00004000	See appendix
	DEF_LAMINATE_END	0x00002000	See appendix
	DEF_LAMI_HOPPER_FULL	0x00001000	See appendix
	DEF_LAMINATE_UNSUPPORTED	0x00000800	See appendix
4	ERR_LAMI_TEMPERATURE	0x00000080	See appendix
	ERR_LAMINATE	0x00000040	See appendix
	ERR_LAMI_MECHANICAL	0x00000020	See appendix
	ERR_LAMINATE_END	0x00000010	See appendix
	ERR_LAMI_COVER_OPEN	0x00000008	See appendix

Word containing *EXT3 type information*

Byte	Bit designation	Mask	Description
1	CFG_EXTENSION_4	0x80000000	Status Extension 4 enabled
2			
3			
4			

Word containing *EXT4 type information*

Byte	Bit designation	Mask	Description
1	CFG_EXTENSION_5	0x80000000	Status Extension 5 enabled
2			
3			
4			

5.2. Configuration of the ESPF server

The ESPF server can be configured using its configuration file. The server settings are taken into account upon server boot.

The configuration file is located in the server setup directory and is named `ESPFSvc.properties`.

This file hosts general server parameters, as well as parameters specific to services hosted on the server.

The following tables describe a selection of such parameters.

Topic	Parameters for the debugging logs
Name	`ESPFService.log`
Value	The `true` value enables the debugging logs (default). The `false` value disables it.
Name	`ESPFService.loglevel`
Value	The following values set the verbosity level of the debugging logs : `1` Fatal `2` Critical `3` Error (default) `4` Warning `5` Notice `6` Information `7` Debug
Name	`ESPFService.logrequest`
Value	The `true` value enables the requests log. The `false` value disables it (default).
Name	`ESPFService.logrequestoutputdir`
Value	Defines the directory for requests log files. (default value is `Tmp`)
Name	`ESPFService.isrelativeoutputdir`
Value	The `true` value indicates that the directory is defined as a relative path to the directory where the service is installed (default). The `false` value indicates an absolute path.

Topic	Parameters for the Named pipe transport layer
Name	`ESPFServerManager.serveraddress`
Value	Base name of the Named pipe channel to be used by the server's clients.
Name	`ESPFServerManager.uniqueid`
Value	Supplementary name for the Named pipe channel to be used by the server's clients. (Name to use = base name + supplementary name)
Name	`ESPFServerManager.disablepipeserver`
Value	The `true` value disables the Named pipe transport layer upon server boot. The `false` value enables it (default).

Topic	Parameters for the Socket transport layer
Name	`ESPFServerManager.port`
Value	States the port number to be used by the server's clients. `18000` is the default value for the EPS.
Name	`ESPFServerManager.enabletcpatstart`
Value	The `true` value enables the Socket transport layer upon server boot. The `false` value disables it (default).

Topic	Parameters for the add-on service
Name	`ServiceAddOnManager.enabled`
Value	The `true` value enables the services upon server boot. The `false` value disables it (default).
Name	`ServiceAddOnManager.addondir`
Value	Defines the directory for hosting the add-on applications that can be executed. (default value is `Tmp\\SrvAddOn`)
Name	`ServiceAddOnManager.isrelativeaddondir`
Value	The `true` value indicates that the directory is defined as a relative path to the directory where the service is installed (default). The `false` value indicates an absolute path.

5.3. Returned errors

The different types of returned errors to a request sent to the server are stated below :

All services	
Code	-32700
Message	(E)Parse error
Description	Invalid JSON was received by the server. An error occurred on the server while parsing the JSON text
Code	-32600
Message	(E) Invalid Request
Description	The JSON sent is not a valid Request object
Code	-32601
Message	(E) Method not found
Description	The method does not exist / is not available
Code	-32602
Message	(E) Invalid params
Description	Invalid method parameter(s)
Code	-32603
Message	(E) Internal error
Description	Internal JSON-RPC Error
Code	-32000
Message	(E) Service not found
Description	The service does not exist / is not available

***CMD* Service**	
Code	1200
Message	(E) Get status error
Description	Get status error
Code	1201
Message	(E) Send command error
Description	Send command error
Code	1202
Message	(E) Reset communication error
Description	Reset communication error
Code	1300
Message	(W) Session already reserved
Description	The communication session is already reserved
Code	1301
Message	(W) Timeout
Description	The send command timeouts
Code	1302
Message	(W) Device not supervised
Description	The target device is not supervised
Code	1303
Message	(W) No Status
Description	The status cannot be read or is invalid

***PRINT* Service**	
Code	1600
Message	(E) Print error
Description	Print Error
Code	1700
Message	(W) Printing session already in progress for the device
Description	The device has already a printing session in progress
Code	1701
Message	(W) Invalid printing session.
Description	The session id sent is not a valid printing session id
Code	1702
Message	(W) Cannot perform action : action already in progress
Description	Cannot perform action: action already in progress
Code	1703
Message	(W) Cannot perform action : setting or bitmap missing
Description	Cannot perform action: setting or bitmap missing
Code	1704
Message	(W) Cannot perform action : processing error
Description	Cannot perform action: processing error
Code	1705
Message	(W) Cannot perform action : job canceled
Description	Cannot perform action: job canceled

***SETTING* Service**	
Code	1400
Message	(E) Get error
Description	Get Error
Code	1401
Message	(E) Set error
Description	Set Error
Code	1402
Message	(E) Import error
Description	Import Error
Code	1403
Message	(E) Export error
Description	Export Error
Code	1500
Message	(W) Invalid session
Description	The session id sent is not a valid session id
Code	1501
Message	(W) Cannot perform action : action already in progress
Description	Cannot perform action: action already in progress
Code	1502
Message	(W) Cannot perform action : import not done
Description	Cannot perform action: import not done

***SUPERVISION* Service**	
Code	1000
Message	(E) Device supervision/un-supervision error
Description	The device cannot be supervised/un-supervised
Code	1001
Message	(E) Set event action error
Description	Set event action error
Code	1100
Message	(W) Device not found
Description	The device cannot be found
Code	1101
Message	(W) Device already supervised
Description	The device is already supervised
Code	1102
Message	(W) Device already un-supervised
Description	The device is already un-supervised
Code	1103
Message	(W) No event waiting for an action
Description	There is no event waiting for an action
Code	1104
Message	(W) Event action to set not found
Description	The event action to set does not exist / is not available
Code	1105
Message	(W) Different event waiting for an action
Description	The event waiting for an action is different

***ADD-ON* Service**	
Code	1800
Message	(E) Launch error
Description	Launch error

***ESPF* Service**	
Code	2000
Message	(E) Get parameter error
Description	Get parameter error
Code	2001
Message	(E) Set parameter error
Description	Set parameter error
Code	2100
Message	(W) The parameter key is invalid
Description	The parameter key is invalid

5.4. Printer states

The printer states are those reported by the Print Center.

5.4.1 Major states

Designation	Description
`OFF`	Unable to communicate
`ERROR`	Printer error (requests a human intervention)
`READY`	Printer Ready
`WARNING`	Next printing unavailable (requests a human intervention)

A `WARNING` state is usually reported before the beginning of a printing but for some specific states it can be reported only when the printing starts.

An `ERROR` state is only reported during a printing.

5.4.2 Minor states

Designation	Description	Major State
`PRINTER_NOT_SUPERVISED`	Not supervised by Evolis Print Center	`OFF`
`PRINTER_OFFLINE`	Printer offline	`OFF`
`PRINTER_STATUS_DISABLED`	Status disabled - Printer on-line	`OFF`
`ERR_BLANK_TRACK`	Magnetic encoding failed Encoding fails on this card, please check the card position in the feeder.	`ERROR`
`ERR_COVER_OPEN`	Cover open error The cover was opened during the printing cycle. Close the cover and click on resume.	`ERROR`
`ERR_FEATURE`	Printing not supported Print cannot be run because it is not supported by the printer	`ERROR`
`ERR_HEAD_TEMP`	Too high temperature Print head temperature too high - wait until it cools down.	`ERROR`
`ERR_HOPPER_FULL`	Output Hopper Full Please remove all the printed cards from the output hopper to resume printing.	`ERROR`
`ERR_MAGNETIC_DATA`	Magnetic encoding failed Invalid data format.	`ERROR`
`ERR_MECHANICAL`	Mechanical error A mechanical error has occurred.	`ERROR`

`ERR_READ_MAGNETIC`	Magnetic encoding failed Magnetic track reading failed.	ERROR
`ERR_REJECT_BOX_FULL`	Reject Box Full Please remove all the cards from the reject hopper to resume printing.	ERROR
`ERR_RIBBON_ERROR`	Ribbon problem The ribbon is cut or stuck to the card.	ERROR
`ERR_WRITE_MAGNETIC`	Magnetic encoding failed Read-after-Write failure.	ERROR
`FEEDER_EMPTY (ERR_FEEDER_ERROR)`	Card feed problem Please check cards, position in the card feeder and gauge adjustment.	ERROR
`INF_WRONG_ZONE_EXPIRED`	Ribbon not valid Compatibility problem between ribbon and printouts credit limit reached. Please contact your reseller.	ERROR
`RIBBON_ENDED (ERR)`	Ribbon end Ribbon end, please replace by a new one.	ERROR
`LAMINATE_END (ERR)`	Film end Film end. Please replace the film.	ERROR
`ERR_LAMINATE`	Film problem Film problem. The film is cut or stuck to the card.	ERROR
`ERR_LAMI_MECHANICAL`	Mechanical error A mechanical error has occurred in the lamination module.	ERROR
`ERR_LAMI_TEMPERATURE`	Temperature error The lamination module encountered a temperature error.	ERROR
`ERR_LAMI_COVER_OPEN`	Door open during lamination The lamination module door got opened during the lamination process. Please close it and retry.	ERROR
`INF_CLEANING`	Regular cleaning required Printer cleaning required. *(this state is reported only when the printing starts, every 50 cards approximatively)*	READY
`INF_CLEANING (INF_CLEAN_2ND_PASS)`	Insert your adhesive cleaning card Please insert your sticky cleaning card. 'Cancel' if you want to proceed with printing. *(this state is reported only when the printing starts, every 50 cards approximatively)*	READY

INF_CLEANING_RUNNING	Cleaning in progress	READY
INF_ENCODING_RUNNING	Encoding in progress	READY
INF_PRINTING_RUNNING	Printing in progress	READY
INF_LAMINATING_RUNNING	Lamination in progress	READY
INF_LAMI_CLEANING_RUNNING	Lamination module cleaning in progress	READY
INF_LAMI_UPDATING_FIRMWARE	Lamination module firmware update in progress	READY
INF_SLEEP_MODE	Printer in Standby mode	READY
INF_UPDATING_FIRMWARE	Firmware update in progress	READY
NOT_FLIP_ACT	Single-sided Printer	READY
PRINTER_READY	Printer ready	READY
INF_RIBBON_LOW	Ribbon close to the end Ribbon close to the end, please proceed with replenishment. *(this state is reported only when the printing starts, every 10 cards approximatively, and changes automatically after 10s)*	WARNING
INF_FEEDER_NEAR_EMPTY	Feeder almost empty The card feeder is almost empty, please refill. *(this state is reported only when the printing starts, every X cards approximatively, and changes automatically after 10s)*	WARNING
BUSY	Printer busy You cannot print while the printer is busy. Please wait or click on "Cancel". *(this state is reported only when the printing starts)*	WARNING
CFG_FLIP (test if not raised)	Single-sided Printer Your single-sided printer cannot print your dual-sided design. *(this state is reported only when the printing starts)*	WARNING
CFG_MAGNETIC (test if not raised)	Magnetic coding option not installed To continue printing without magnetic coding click on resume. *(this state is reported only when the printing starts)*	WARNING
CFG_EXTENDED_RESOLUTION (test if not raised)	Incompatible parameter This resolution parameter is not compatible with this printer / ribbon. *(this state is reported only when the printing starts)*	WARNING
DEF_CARD_ON_EJECT	Remove card Remove the card from the manual feeder.	WARNING

DEF_COOLING	Cooling in progress Printer cooling in progress.	WARNING
DEF_COVER_OPEN	Cover open Close your printer cover.	WARNING
DEF_HOPPER_FULL	Output Hopper Full Please remove all the printed cards from the output hopper to resume printing.	WARNING
DEF_NO_RIBBON	No ribbon Replace the ribbon.	WARNING
DEF_PRINTER_LOCKED	Communication with the printer is locked Contact your dealer	WARNING
DEF_UNSUPPORTED_RIBBON	Ribbon incompatible with this printer model The ribbon inserted cannot work with this printer model.	WARNING
DEF_WAIT_CARD	Waiting for a card insertion Please insert your card manually.	WARNING
ERR_BAD_RIBBON	Ribbon installed is incompatible with settings The ribbon installed does not correspond to the manuallly defined settings. Printing cannot take place. *(this state is reported only when the printing starts)*	WARNING
FEEDER_EMPTY (DEF)	Card feed problem Please check cards, position in the card feeder and gauge adjustment.	WARNING
INF_CLEANING_ADVANCED	Advanced cleaning required Printer advanced cleaning is required. *(this state is reported only when the printing starts, every 50 cards approximatively)*	WARNING
INF_CLEANING_LAST_OUTWARRANTY	Regular cleaning mandatory Click on 'Cancel' and proceed with cleaning immediately. Would you continue, this will void the print head warranty. *(this state is reported only when the printing starts)*	WARNING
INF_CLEANING_REQUIRED	Regular cleaning mandatory - No card issuance allowed by your Administrator Click on 'Cancel' and proceed with cleaning immediately. *(this state is reported only when the printing*	WARNING

	starts)	
`INF_UNKNOWN_RIBBON (1)`	Ribbon not identified Ribbon identification impossible. Please proceed with Manual settings.	`WARNING`
`INF_UNKNOWN_RIBBON (2)`	Ribbon not identified Ribbon identification impossible. Please proceed with Manual settings.	`WARNING`
`INF_WRONG_ZONE_ALERT`	Ribbon not valid Compatibility problem between ribbon and printer. Less than 50 printouts remaining. Please contact your reseller. *(this state is reported only when the printing starts, every 10 cards approximatively)*	`WARNING`
`INF_WRONG_ZONE_RIBBON`	Ribbon not valid Compatibility problem between ribbon and printer. Please contact your reseller. *(this state is reported only when the printing starts, every 25 cards approximatively)*	`WARNING`
`RIBBON_ENDED (DEF)`	Ribbon end Ribbon end, please replace by a new one.	`WARNING`
`DEF_NO_LAMINATE`	No film No film in lamination module. Please replace the film.	`WARNING`
`INF_LAMINATE_UNKNOWN`	Film not identified Unknown film. Please contact your reseller.	`WARNING`
`INF_LAMINATE_LOW`	Film close to the end Film close to the end. Please arrange for replacement. *(this state is reported only when the printing starts, every 10 cards approximatively, and changes automatically after 10s)*	`WARNING`
`LAMINATE_END (DEF)`	Film end Film end. Please replace the film.	`WARNING`
`DEF_LAMINATE_UNSUPPORTED`	Incompatible film Film incompatible with lamination module. Please contact your reseller.	`WARNING`

DEF_LAMI_COVER_OPEN	Door open Lamination module door open. Close the lamination module door.	WARNING
INF_LAMI_TEMP_NOT_READY	Adjusting temperature Lamination module temperature adjustment in progress. Please wait…	WARNING
DEF_LAMI_HOPPER_FULL	Output jammed The lamination output is jammed. Please remove the card(s) and retry.	WARNING

5.5. Document history and versions

Date	Version	Description
30/03/2015	A0	Initial version
23/04/2015	A1	Revision translation
10/06/2015	A2	Update for public release
04/11/2015	A3	Update Binary status of a printer and Printer states
23/05/2016	A4	Update Binary status of a printer and Printer states
12/06/2017	A5	Update SUPERVISION service, Binary status and Printer states appendix