Manual

on

CPR services

Annex 5

Logon and general use of CPR services, programming guidelines

CPR Office
Datavej 20, PO Box 269, 3460 Birkerød, Denmark
Emailcpr@cpr.dk. Telefax +45 45 82 51 10 Website: www.cpr.dk

Document information

Title:	**Manual on CPR services Annex 5** **Logon and general use of CPR services, programming guidelines.**
Project:	
Location:	O:\GTS\CPR\Systembeskrivelse\Serviceshåndbog\SERVICEHÅNDBOG-Bilag_5.DOC
Entry into force:	Immediately
Author:	Peter N Bilby
Approved by:	The CPR Office: Jørgen Ø. Møller by signature on covering note CSC: Bo Jystrup by signature on covering note
Distribution:	
Signed on:	18 August 2006

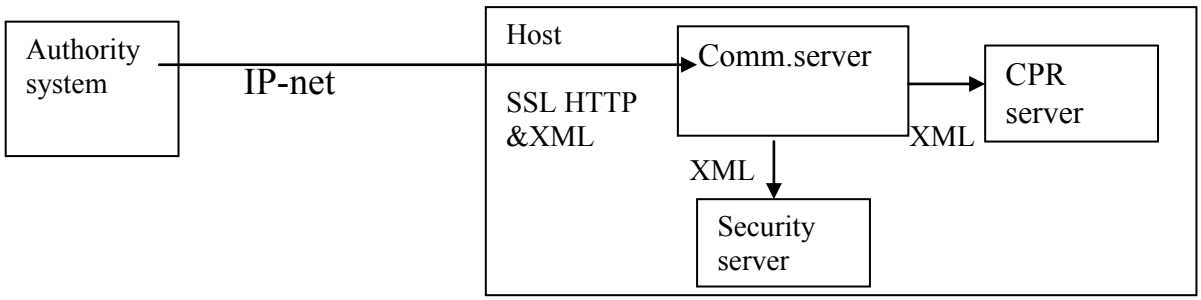
Version	Date	Changed pages or sections	Comments
001	9 February 2001	All	First publication
002	15 June 2001	All	XML implemented
003	23 May 2002	All	XML SSL incorporated
004/017	20 December 2002	Section 4	Incorporated Possibility of reusing sockets (Keep-Alive)
005/023	10 March 2004	Section 3	Change of portal number to 683
006/045	18 August 2006	All	Adapted to HTTP 1.1

Table of contents

1. Introduction

The following section describes how the client establishes connection, and logs on to the CPR server.

2. Overview



Communication

The client's communication of data in XML format with CPR services is through a communications server on the host which, like the client, is connected to the IP network.

Non-disclosure SSL

Communication is based on an SSL socket connection.
The following are supported on the host side:
Encryption method for key exchange: RSA 1024 bit
Hash method to secure integrity: SHA1
Encryption method for non-disclosure of beneficial data: DES 56 bit

Headers

HTTP data comprises a header for the following data in XML.

On the IP network, there is an additional IP header and a TCP header in front of the HTTP header.

IP	TCP	HTTP	XML

IP

The IP header is used to create a connection between a client and a communication server on the host, which gives access to the CPR server and the Security Server. In the IP network, the host is identified by its IP address, and the communications server listens to one of the host's ports.

HTTP

HTTP data is used for the client to communicate with the communications server's own services regarding:

- Choice of service (Security Service Logon and CPR services).
- Redirection of IP address and port number.
- Error messages concerning communication.

XML in connection with Logon

The XML format is used by the Security Server, e.g. in connection with Logon and communication with the CPR.

Set of characters

Note that the communications server converts between the ISO-8859-1 set of characters on the network and EBCDIC 277 so that only the characters which are common for these two sets of characters can be processed by the other parts of the server. If other characters are received, this will result in an error message.

- In practice, a PC can be configured to use the set of characters cp=850, but only the characters common with ISO-8859-1 and EBCDIC 277 may be used.

3. Establishment of a socket connection

Calling the CPR server

Before dialogue is established, a socket connection must be established over the IP network between the client and an environment behind the CSC communications server.

Environment	<IP-name>:<Port>
Production	prod.ajou.cpr.dk:683
Demo	demo.ajou.cpr.dk:681
External test	xtst.ajou.cpr.dk:682

In the CSC name server, all IP names point towards the IP number 147.29.11.10,
which is the external operating environment of the CSC.

Logon and each subsequent transaction uses its own socket connection.

4. HTTP communication

4.1 Request

	The HTTP header consists of: - Start line Request and response are structured differently. - Communication lines Request and response have the same form. - End line Request and response have the same content. All lines in the HTTP header include the last two characters CRLF (HEX '0A0D') written here as ↵.

Start line

WORD	**DESCRIPTION**
<Type of request>	Must always be POST
<Blank>	ASCII character blank
<Path for service>	Service Path for service Logon: /cics/dmwg/cscwbsgn CPR-Application: /cpcacpra/ajou/xyz
<Blank>	ASCII character blank
<Protocol>/<Version>	<Protocol> Must always be HTTP <Version> Must be at least 1.0
<CRLF>	HEX '0A0D'

Communication lines

Linieident	**DESCRIPTION**
Host:	as line value: Host's internet address. I.e. Either: xtst.ajou.cpr.dk Or: prod.ajou.cpr.dk In the HTTP 1.1 version, there must be a blank space after special characters. E.g.

	Host**:** prod.ajou… Blank space after colon.
User-Agent:	Contained as line value: Applikationsnavn/Majorciffer.minorciffer. This information has been agreed with the CPR. Must be present. (Remember blank space after:)
Content-Length:	Contained as line value: Length in bytes of the data in XML format following the HTTP header. Must be present. (Remember blank space after:)
Cookie:	Contained as information: TOKEN=ZZZAAAAAAAA Content is therefore not validated in connection with Logon. Upon request of application, the value equals the token received in connection with Logon. Syntax for the value of the token which is received in connection with Logon is (11 pos.) 3 Zs and 8 small letters. Must be present. However not in connection with Logon (Remember blank space after:)

End line

The end line only includes the value ↵

Like all other lines, the last communication line is also concluded with ↵
The last bytes in the header are therefore HEX ’0A0D0A0D’

XML

After the end line in the HTTP header follows the XML part. This consists of an XML header, any command statements as well as the block with information about name space (xmlns) where the gctp block is located.

Example

```
POST /cics/dmwg/cscwbsgn HTTP/1.1↵
Host: prod.ajou.cpr.dk
User-Agent: CPR/1.0↵
Content-Length: 420↵
Cookie: Token=ZZZXXXXXXXX↵
↵
  <?xml version="1.0" encoding="ISO-8859-1"
standalone="yes"?>
<root xmlns="http://www.cpr.dk"><Gctp v=”1.0">
…
</Gctp></root>.
```

NOTE! CPR clients type the XML string as a line without ↵ and extra blank spaces. As XML is used, others are, however, not required to

follow the same rule.

4.2 Response

As a reply to the request, a response is returned. The response is structured in much the same way as the request.

Note that there must be a reaction to e.g. redirection.

Start line in response

WORD	**DESCRIPTION**
<Protocol>/<version>	<Protocol>: HTTP <Version>: E.g.: 1.1. Host's version.
<Returncode>	HTTP return code if the communication of the request with the communications server went well (200) otherwise error. The standardised return codes in HTTP are used, as well as supplementary return codes on Logon, as seen in section 3.2.1.
<Returntext>	Attached standardised HTTP return text. In connection with return code 200, the return text is: OK
<CRLF>	HEX '0A0D'

Note that a small but random number of blank spaces may come after a word.

Communication lines in response. May occur in other sequences than shown and not all have to occur.

Linieident	**DESCRIPTION**
User-Agent	Structured as requested.
Content-Length	Structured as requested. I.e. Contained as line value: Length in bytes of the data in XML format following the HTTP header.
Content-Typ	Content-Typ :text/xml
Pragma	Pragma: no-cache
Date	Date: day, dd month, yyyy hh:mm:ss Timezone Date selected due to the update of other programs.
Expires	Expires: day, dd month, yyyy hh:mm:ss Timezone Date selected due to the update of other programs.
Cookie	The structure enables the supply of information about token and redirection from the host. Such information is only present in the line if the value

	is new or when it is changed. Structure of the line: Set-Cookie:<Informationlist > Either <Informationlist >::=<Information> or <Informationlist >::= <Information>; <Informationlist > Token: <Informationlist >::=Token=<token;Path=/ If <token> = ZZZzzzzzzzz, token is not accepted by the host's security system. Redirection: <Information> ::= Ipaddr=<ipaddress> <Information>::=Port=<port> <Information> ::=Path=<path for application>
Through:	Through:<Name of proxyserver> <Name of proxyserver>::=HTTP/1.0imainframeweb.csc.dk (IBM HTTP Server)
Connection:	Connection: Keep-Alive Only upon reciept of such message may the client attempt to reuse the used socket. There is no guarantee that the host maintains knowledge of this socket when the client tries to use it. For some applications which do not use Keep-Alive correctly, it may be possible to add the HTTP header"`Connection:close`"
Proxy-Connection:	Proxy-Connection : Keep-Alive
<CRLF>	HEX '0A0D'

End line

The end line only includes the value ↵

Like all other lines, the last communication line is also concluded with ↵
The last 4 bytes in the header are therefore: HEX '0A0D0A0D'

Example

```
HTTP/1.1  200 OK↵
Pragma: no-cache↵
Date:Mon, 21 Mar 2002 15:31:31 GMT↵
```

```
Content-Length:2443↵
Content-Type: text/xml ↵
Expires: Mon, 21 Mar 2002 00:00:02 GMT↵
↵
```

Below are the above-mentioned 2443 byte data in XML format

4.2.1 Data in XML format in connection with Logon

XML documents must be initiated with an XML header. Following this there may be command statements with URL for XML Schema or DTD describing the structure and its data. Subsequently, a block follows with information about ownership of the GCTP block and its data. The GCTP block is located within the block on ownership.

Example

```
<?xml version="1.0" encoding="ISO-8859-1"
standalone="yes"?>
<root xmlns="http://www.cpr.dk">
   <Gctp v="1.0">
      …
   </Gctp>
</root>
```

Return codes

The CPR has defined return codes from the host's security system. These are used in XML data.

returnCode	**Text**
900	Signon successful
901	Token not known
902	User ID not defined in the security system
903	User ID inactive in the security system
904	Invalid User ID entered
905	Invalid password entered
906	Your password has expired
907	Both passwords must be the same
908	New password not valid
999	Implementation error

Ordinary Logon

To log on to the system, the user ID and password must be sent as data in XML format with a POST request.

Example

```
POST /cics/dmwg/cscwbsgn HTTP/1.1↵
User-Agent: CPR/1.0↵
Content-Length: 75↵
↵
<?xml version="1.0" encoding="ISO-8859-1"
standalone="yes"?>
<root xmlns="http://www.cpr.dk"><Gctp
v=”1.0”><Sik function= "signon" userid="x…x"
password="xxxxxxxx"/></Gctp></root>
```

Change of password

NOTE! CPR clients type the XML string as a line without ↵ and extra blank spaces. As XML is used, others are, however, not required to follow this rule.

Example

When a user changes password, the HTTP part is as above, whereas data in XML format is as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"
standalone="yes"?>
<root xmlns="http://www.cpr.dk"><Gctp
v=”1.0"><Sik function="newpass"  userid=”x…x”
password=”xxxxxxxx” newpass1=”xxxxxxxx"
/></Gctp></root>
```

Response when things go well

Example of response from the server after Logon and Change of password:

```
HTTP/1.1  200 OK↵
Content-Length:64↵
Expires: 0↵
Pragma: no-cache↵
Content-Type: text/xml ↵
Set-Cookie: Token= ZZZabcdefgh; Path=/↵
↵
<?xml version="1.0" encoding="ISO-8859-1"
standalone="yes"?>
<root xmlns="http://www.cpr.dk"><Gctp
v=”1.0"><Sik><Kvit r= ”returKode” t=”Signon
```

Missing token

```
udført" v="900"/></Sik></Gctp></root>
```

If a token is not forwarded in the request (after Logon), or if an expired token is forwarded in the request, the following data may e.g. be returned in XML format:

```
<?xml version="1.0" encoding="ISO-8859-1"
standalone="yes"?>
<root xmlns="http://www.cpr.dk"><Gctp
v="1.0"><Sik><Kvit r="returKode" t=" Token
unknown" v="901"/></Sik></Gctp></root>
```

The HTTP error code is still "200 OK" as the communication itself went well.