DIGITAL SUNDHED
SAMMENHÆNGENDE DIGITAL SUNDHED I DANMARK

SOSIGW

- **Programmers Guide for SOSIGW 1.2**

Indeks

1 Revisionshistorik

Version	Dato	Ændring	Ansvarlig
1	09/10/09	Initiel version af dokument	jre
2	29/08/13	Segmentering af cache fra version 1.2	CHE
3	1/11/13	Rettet svn link	KPN
4	10/6/14	Tilføjet logoutWithResponse	KPN

2 Introduktion

Dette dokument er en kort introduktion til SOSI-GW API'et, og brugen af dette.

Vejledningen er oprindeligt udarbejdet på engelsk, og dette er fastholdt efterfølgende.

3 Prerequisites

- Web-services using SOAP
- WS-Addressing

3.1 Deployment of SOSI-GW in your own infrastructure

The SOSI-GW can be deployed in your own infrastructure if you wish to have full access to log files and configure the SOSI-GW for a specific purpose. The Programmer's Guide to SOSI-GW focuses on the *use* of SOSI-GW, and further information on the SOSI-GW (including deployment instructions) is described in the documents

- Installationsvejledning
- Installationsvejledning NSP
- Driftsvejledning
- Brugerdokumentation

These documents can be requested from the *NSP operatør* (http://www.nspop.dk).

After deployment the web-services can be found either at

http://<HOSTNAME>:8080/sosigw/service/sosigw

for the full webservice, or at

http://<HOSTNAME>:8080/sosigw/service/sosigw-restricted

for the bootstrap webservice.

3.2 Use of national SOSI-GW installations in the production environment

In the production environment the SOSI-GW is deployed as part of the NSP infrastructure (National Service Platform). Organisations with a local dNSP installation (presently only the Regions) have their ”own” SOSI-GW. All other organisations use a shared SOSI-GW deployed in a central location as a part of the cNSP installation. For more information on the NSP-concept please refer to the *NSP operatør* (http://www.nspop.dk).

3.3 Use of national SOSI-GW installations in the test environments

In most cases use of the national SOSI-GW installations will be a better choice than your own installation, as the SOSI-GW de facto is an infrastructure component that should be considered a ”black box”.
As from early 2013 four national test environments[1] are available for test purposes and education of users. Each of the test environments has two SOSI-GW installations, one for the dNSP, and one for the cNSP.
The *NSP operatør* is responsible for configuration and maintenance of the environments, and after signing up with the *NSP operatør* for use of the environments all parties have access to the SOSI-GW API of each of the installations.

4 Using SOSI-GW

You must be able to build WS-requests with a valid level 1 UserIDCard, in the SOAP-Header.

This is required for **all** requests to and through SOSI-GW. From SOSI-GW version 1.2 and forth it is possible to use IDCards with a level different than 1. Please refer to section 6.2 for more information.

You can either implement the signing in your application, or leave it to the gateway to do so using a browser. The two approaches are described in detail in the following.

4.1 Application-integrated signing of IDCard

For the application-integrated signing case, the normal flow of requests is as follows:

[1] for more information on the *fælles testmiljøer* please refer to https://www.nspop.dk/pages/viewpage.action?pageId=8915030

Operation	Description	Details
requestIdCardDigestForSigning	This is the "login" step.	In return you get a digest that you can sign, if you wish to perform this step in your client. Also returned is a URL that, when opened in a browser, starts an applet that performs the signing for you. Use either of them. Calling requestIdCardDigestForSigning sets the IDcard-state to "unsigned" so that no calls may go through "proxy".
getValidIdCard	This call enables your application to check if there is a signed card for the user in the gateway	A fault is returned if there is no card, or if the card is present, but unsigned. The purpose is to enable an application to poll while waiting for the browser-based signing to take place. If there is a signed card, it is returned – however with its digtial signature removed. The purpose of this is to let the application inspect attributes such as the remaining period of validity.
signIdCard	Used in conjunction with requestIdCardDigestForSigning when not using browser-based signing.	As input, supply the signed digest and the certificate used to perform the signing. The gateway then contacts the STS and stores the resulting IDcard in the cache.

Operation	Description	Details
proxy	This is the gateway operation, used to relay messages to other parties. Call it as many times as needed.	Note that this operation is not in the wsdl for SOSI-GW. By default it lives on http://localhost:8080/sosigw/proxy/soap-request. If there is a valid, signed IDCard in the gateway, it relays the message to the destination provided in the WS-Addressing headers. If there is no signed IDCard in the gateway, a fault is returned, with the same information as is returned by requestIdCardDigestForSigning, but provided in a header.
logout	When you want the cached IDcard to be invalidated.	You should probably call this when the user logs out of your system, but not necessarily, as not doing so provides a kind of single sign on to the services used through the gateway.
logoutWithResponse	When you want the cached IDcard to be invalidated.	Same as above, but returns ”ok” on succesfull logout or returns a DGWS exception

4.2 Browser-based signing

For the browser-based signing case, the normal flow of requests is as follows:

Operation	Description	Details
requestIdCardDigestForSigning	This is the "login" step.	Use the URL returned to start a browser session.
getValidIdCard	This call enable your application to check if there is a signed card for the user in the gateway.	Call at regular intervals, probably no more than once a second or so, to know when the browser based signing has been completed. You will likely want to allow your user to redo the process, if the user has closed the browser by mistake or for some other reason wants to restart the signing process.
signIdCard	- not used -	Not used for browser based signing, as this is handled by the gateway and applet.
proxy	This is the gateway operation, used to relay messages to other parties. Call it as many times as needed.	Same as for application-based signing (see above).
logout	When you want the cached IDcard to be invalidated.	Same as for application-based signing (see above).
logoutWithResponse	When you want the cached IDcard to be invalidated.	Same as above, but returns ”ok” on succesfull logout or returns a DGWS exception

4.3 Use of the ”implicit login” approach

It is possible to completely ignore the existence of **requestIdCardDigestForSigning** by just skipping it and calling **proxy** and inspecting the error code returned by it.

If the error code is "sosigw_no_valid_idcard_in_cache", there is no UserIDCard in the SOSI-GW server for the current user (or, rather, current NameID in the SOAP header)

In this case, SOSI-GW returns the same response as **requestIdCardDigestForSigning** would have done, except it is returned in the SOAP header rather than the body of the response, as the body has to be a fault element.

Please refer to "sosigw_implicitLoginHeader.xsd" for the schema describing the response.

4.4 Fault codes

The complete set of fault codes returned by SOSI-GW is listed here.

But please note that it is quite possible to get other fault codes, as faults from STS and faults from the proxied calls are also returned. Please refer to the documentation of the STS and the services called through the SOSI-GW in order to get the complete set of possible fault codes.

Fault code	Description
sosigw_no_valid_idcard_in_request	Returned when there is no UserIDCard in the SOAP header
sosigw_missing_signinginfo_in_request	Returned when the key-info or signed digest is missing from the request to signIdCard.
sosigw_syntax_error_in_request	Returned when parsing the headers failed.
sosigw_awaiting_signing	Returned by getValidIdCard when there is a card in the cache, that has not been signed yet.
sosigw_no_valid_idcard_in_cache	Returned when there is no IDCard in the cache that matches the nameID of the input.
sosigw_internal_error	Generic error indication an internal unexpected failure. Please refer to the debug-log on the server in this case.
sosigw_proxy_error	A problem occurred trying to connect to the remote host or delivering the request to the remote host while proxying a request.
sosigw_access_denied	Client or service is not authorized by the white-lists.

5 The bootstrap webservice

The bootstrap webservice exposes only the **requestIdCardDigestForSigning** and signIdCard handles, but unlike the full webservice, the bootstrap webservice is not ip-whitelisted, nor does it use a shared secret, if such is configured.

6 Use of SOSI-GW in a shared environment

From version 1.2 and forth the SOSI-GW can be deployed in a shared environment where more than one organisation can use the services of the SOSI-GW. Until version 1.2 this has not been possible due to security restrictions (risk of unintentional or malevolent cross-organisational access to the SOSI-GW cache of signed IDCards).
In a shared environment the SOSI-GW and the surrounding infrastructure[2] partition the IDCard cache, ensuring that each organisation only has access to its own part of the cache.
This has been implemented in a way that does not affect the existing SOSI-GW API, and existing use of the SOSI-GW as well as existing documentation is unchanged.

6.1 Use of ”PassThrough” header

In the shared environment all requests go through the SOSI-GW. Some requests should pass through the SOSI-GW unchanged (i.e. no processing of the request, including evaluation of the IDCard), and in order to allow this a ”PassThrough” SOAP header has been introduced from version 1.2 which is placed in the SOSI-GW namespace (http://sosi.dk/gw/2007.09.01):

```
<sosigw:PassThrough />
```

Requests with this header will be stripped of the header, and will otherwise be left unchanged, and passed through the SOSI-GW to the specified destination.

6.2 Use of signed IDCards in calls to the Proxy interface

With the SOSI-GW placed before the DCC, new usage scenarios emerge, and the Proxy interface has been modified in order to let calls containing signed IDCards pass through unchanged. This includes level 2 IDCards (username+password), even though they are not technically ”signed”.
The SOSI-GW now accepts all IDCard levels, and the way requests are processed is as follows for the Proxy interface:

Calls that are passed through without exchange of IDCard:

- Requests with IDCard level 2: The request is passed through the SOSI-GW
- Requests with Signed IDCard level 3 and 4: The request is passed through the SOSI-GW

All other calls are processed by the SOSI-GW:

- Requests with IDCard level 1
- Requests with unsigned IDCard level 4

[2] The infrastructure components implied here are managed by the *NSP-operatør* in the production environment and the *nationale testmiljøer*.