**Origination Date:** 11/15/2002

**Originator:** Bellsouth

### Change Order Number: NANC 372

**Description:** SOA/LSMS Interface Protocol Alternatives

**Functionally Backward Compatible:** No

## IMPACT/CHANGE ASSESSMENT

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| FRS | IIS | GDMO | ASN.1 | NPAC | SOA | LSMS |
| Y | Y | N | N | Y | Y | Y |

**Business Need:**

Currently the only interface protocol supported by the SOA-to-NPAC interface and NPAC-to-LSMS interface is CMIP. The purpose of this change order is to request analysis be done to determine the feasibility of adding other protocol support such as CORBA or XML. The primary reasons for looking into a change would be, 1.) Performance, and 2.) Implementation complexity.

**Description of Change:**

Dec ’02 LNPAWG, after a brief introduction, the group agreed to discuss this change order in January ’03 in the new Architecture Planning Team meeting.

**Jan ’03 APT**, discussion:

The team began with a discussion on the CMIP Alternative Business Need in order to determine if we need to improve CMIP or identify an alternative.

* Dave Cochran, BellSouth and the originator of NANC Change Order 372, discussed potential drivers and cited:
* Cost of maintaining internal CMIP interface expertise and resources
* Ability to take advantage of in-house expertise for some of the newer architectures, e.g., CORBA, XML, JAVA, J2E
* It was stated that CMISE was considered a reasonable protocol for managing network elements in the mid-1990s due to its flexibility.
* LNP rules include encryption/decryption functionality. We need to discuss authentication and associated issues.
* It was mentioned that if lowering the level of encryption is identified as a benefit for a new protocol, we should also consider that for CMIP.
* CMIP is a very robust protocol for describing and managing network elements, but where that robustness begins to become burdensome is subjective.
* We need to keep in mind that we need a real-time interface.

**Feb ’03 APT**, discussion:

Dave Cochran, BellSouth, will be providing more input (business drivers, data, operational feedback, etc.) to facilitate further discussion. Sub-tasks still need to be prioritized.

**Dec ’03 APT**, discussion:

No further discussion at this time. Leave off list of change orders discussed during the APT meeting.

**Jan ’07 APT**, discussion:

The APT was activated during the Nov ’06 LNPAWG meeting. No discussion on alternative interfaces took place during that meeting, but change orders (including 372) were reviewed during the Jan ’07 meeting. The brief discussion included: *CMIP-to-XML/SOAP -- It was asked if there is a business need to transition from CMIP to XML/SOAP. It was suggested that since we are tunneling XML into CMIP, we should explore the future evolution of the interface. Service Providers are to discuss internally any drivers for moving from CMIP to XML/SOAP for the SOA and LSMS interfaces including the impact of increasing the size of messages*.

**Mar ’07 APT**, discussion:

More discussion took place regarding an additional NPAC interface using XML/SOAP. For the May ’07 meeting, Service Providers and vendors are to bring any additional data or information to share with the group.

**May ’07 APT**, discussion:

1. The IT industry is generally moving towards an XML/SOAP interface. However, there are performance issues and questions. Message size would be greatly increased. Need to investigate compression capabilities.

2. It will be worth pursuing for the long term. Not sure what is next step. Need to find a business driver for pursuing this.

3. The WICIS transfer is planning on implementing a flash-cut to XML (Sep ’08). Plan is to continue to support CORBA interface for testing purposes only. Keep this in mind when planning the NPAC implementation.

4. The group will discuss more during the Jul ’07 mtg, including pros/cons analysis, LOE, and any input on the business case.

**Jul ’07 APT**, discussion:

1. In response to May ’07 #3 above, a question was asked about the ATIS decision to move WICIS from CORBA to XML/SOAP. It was explained that the major driver for the ATIS recommendation was to consolidate the various systems onto a single interface type (XML/SOAP), and not necessarily specific to WICIS. It was also mentioned that the NPAC would be supporting two interface types by adding XML/SOAP, since both CMIP and XML/SOAP would need to be supported on the NPAC for the foreseeable future. Sunsetting of the CMIP interface (and only having the XML/SOAP interface) was briefly discussed, but it was also mentioned that the industry has never sunset any previous NPAC functionality.

2. All Service Providers will investigate internally whether or not their companies are moving towards XML/SOAP, and whether or not they support the ATIS position of consolidating interface types towards XML/SOAP. This will be discussed at the Sep ’07 meeting, to gauge industry interest in developing an XML/SOAP interface for the NPAC.

**Sep ’07 APT**, discussion:

1. Deb Tucker, VZW, provided the historical info (from multiple ATIS documents) for ATIS and the single interface item. The current situation for most Service Providers is that new systems are going with XML and legacy systems stay on their existing protocols based on each company’s cost/benefit analysis. The group agreed to continue to discuss this item in future meetings. From the NPAC perspective, support for both interfaces is required since a flash cut cannot be assumed.

2. Given the APT’s charter, the correct way to look at this change order is from an architecture perspective. Several items to consider: messaging (continue to use a session approach like CMIP, or an approach like web-services where it’s set up then broken down when the message is done?), security (how does it change with a web services approach?), message content/architecture (same messages used today with CMIP will be used for XML?), performance/message compression, business rules/error handling, efficiencies in data model (e.g., having DPC at the LRN level), audits (the effect on large messages).

3. Business Case. Need to get to the point where the group can either build or not build a strong business case. May need a document to define an XML/SOAP interface which would help answer the question on the business case. Security will be the first issue discussed at the Nov ’07 meeting.

**Nov ’07 APT**, discussion:

1. The wireless group has been discussing this. They will summarize their recent discussion, and forward some relevant bullet points on to the Architecture team. These bullet points will be used as starting point discussions.

2. The group will further discuss dedicated link versus VPN (http/https. Private network/public network), IP security, .data security (encryption).

**Mar ’08 APT**, discussion:

Wireless service providers may have additional input after WICIS 4.0 implementation in Sep ’08.

**Sep ’11 APT**, discussion:

Discussion began again about moving to a different protocol (e.g., XML) in the NPAC, as this could be a driver to move to support IPv6. The group agreed to review 372 and come to the November meeting prepared to discuss.

**Nov ’11 APT**, discussion:

The group reviewed the following slide deck, and began more detailed discussions.



**Jan ’12 APT**, discussion:

As part of our ongoing discussion on NANC 372 – Alternate NPAC Interface, Neustar agreed to put together a list of questions to assist providers with discussions within your company. As part of Action Item 110911-APT-02 please review these internally and provide responses for our NANC 372 discussion in the January 2012 LNPA WG APT meeting.

Areas where decisions need to be made by LNPA WG:

1. Should the interface protocol be SOAP or HTTPS?
2. Should the interface data encoding be XML or JSON?
3. Should the interface be connection-oriented or connection-less?
4. Should the interface be session based (like the CMIP interface) or single request (like most web traffic)?
5. Should this be a push interface (like the CMIP interface) or should it be a pull/poll interface where providers ask the NPAC if there are any new transactions/messages for them?
6. Should the interface security be a digital signature (like CMIP) or HTTPS where the entire message is encrypted including client authentication?
7. Should recovery of missed data be SWIM based (like CMIP) or should the NPAC constantly attempt to send until successful delivery?
8. How can create/modify/delete notifications be enhanced to make them more efficient?

Current working assumptions:

1. SOA functionality will be implemented.
2. LSMS functionality will be implemented.
3. The interface protocol will be HTTPS.
4. The data encoding will be XML.
5. The interface will be connection-less.
6. The interface will be session-less based (authentication on each request).
7. The interface will push messages in real time.
8. Security will be HTTPS where NPAC generated keys are distributed to SOAs/LSMSs.
9. Recovery will be enhanced to deliver messages until successful.
10. Notifications will be enhanced for efficiency.

After the Jan ’12 APT and in preparation for the Mar ’12 APT, the following was added to document the discussion. Discussion and updates from the Mar ’12 APT meeting, pro/con descriptions.

Interface Protocol – include SOAP envelope or use just straight HTTPS (XML/JSON). Current Working Assumption: interface protocol will be HTTPS (XML/JSON).

Given today’s computing environment, an interface protocol using HTTPS is the working assumption because it is widely used today. The extra step of using a SOAP envelope is not necessary.

HTTPS (XML/JSON)

Pro – widely used today on the internet (with secure applications like online banking), smaller message, simplified by not using SOAP wrapper.

Con – over-all packet size is not as compact as a binary protocol (e.g., CMIP).

SOAP

Pro – widely used today on the internet.

Con – over-all packet size is not as compact as a binary protocol (e.g., CMIP), extra step of using a SOAP wrapper within HTTPS is not considered necessary, extra step uses more system resources, extra step requires more development, synchronous so NPAC and SOA/LSMS would need both Client and Server.

Data Encoding – XML or JSON. Current Working Assumption: data encoding will be XML.

XML is widely used throughout the software industry and people resources are readily available. XML has gone through years of standardization and it uniquely provides standards-based solutions for cases that deal with extensibility, digital signing, and data encryption. XML is a good choice for native data representation for the NPAC. This addresses one of the business needs of this change order which is to minimize implementation complexity. JSON is the newest technology. However, since it is newer, there are not as many development tools available nor is it as widely known.

XML

Pro – widely used today, people resources readily available, less implementation complexity, wide variety of development tools available, very rich syntax that allows for expression of complicated data structures.

Con – not cutting-edge technology, longer parsing time, verbose.

JSON

Pro – newest technology, less complex so faster parsing time, less restrictive data interchange protocol, smaller packet size, more readable.

Con – lacks standardization, less mature and not as well known as XML, fewer development tools available, fewer people resources available, fewer production implementations than XML, less rich syntax limiting expression of data structures.

The following is a comparison of the NPAC Service Provider objects defined in XML and JSON (assume just ID, name, and type). This demonstrates that XML is more clearly and more specifically defined than the JSON syntax definition/specification (JSON definition is descriptive only).

* 1. SPID – XML is defined as a four byte string. JSON is just a string.
  2. Name – XML is defined as up to a 40 byte string. JSON is just a string.
  3. Type – XML is restricted to one of six defined values. JSON is just a string.

XML schema:

        <xs:simpleType name="ServiceProvId">

                <xs:restriction base="xs:string">

                        <xs:length value="4"/>

                </xs:restriction>

        </xs:simpleType>

        <xs:simpleType name="GraphicString40">

                <xs:restriction base="xs:string">

                        <xs:maxLength value="40"/>

                </xs:restriction>

        </xs:simpleType>

        <xs:simpleType name="ServiceProvName">

                <xs:restriction base="GraphicString40"/>

        </xs:simpleType>

        <xs:simpleType name="ServiceProvType">

                <xs:restriction base="xs:token">

                        <xs:enumeration value="wireline"/>

                        <xs:enumeration value="wireless"/>

                        <xs:enumeration value="non\_carrier"/>

                        <xs:enumeration value="class1Interconnected"/>

                        <xs:enumeration value="sp\_type\_4"/>

                        <xs:enumeration value="sp\_type\_5"/>

                </xs:restriction>

        </xs:simpleType>

        <xs:complexType name="ServiceProvNetworkData">

                <xs:sequence>

                        <xs:element name="service\_prov\_id" type="ServiceProvId"/>

                        <xs:element name="service\_prov\_name" type="ServiceProvName" minOccurs="0"/>

                        <xs:element name="service\_prov\_type" type="ServiceProvType" minOccurs="0"/>

                </xs:sequence>

        </xs:complexType>

The following is a sample of the XML data segment:

        <...>

                <service\_prov\_id>X109</service\_prov\_id>

                <service\_prov\_name>Service Provider/3</service\_prov\_name>

                <service\_prov\_type>non\_carrier</service\_prov\_type>

        </...>

JSON syntax definition/specification:

        ServiceProvId           service provider identifier as a JSON String of 4 characters

        ServiceProvName         service provider name as a JSON String of maximum 40 characters

        ServiceProvType         service provider type as a JSON String with the following possible values

                                        wireline

                                        wireless

                                        non\_carrier

                                        class1Interconnected

                                        sp\_type\_5

                                        sp\_type\_6

        object

        {

                ServiceProvId   service\_prov\_id ;

                ServiceProvName service\_prov\_name ; [ OPTIONAL ]

                ServiceProvType service\_prov\_type ; [ OPTIONAL ]

        }

        ServiceProvNetworkData;

The following is a sample of the JSON data segment:

        {

        ...

                "service\_prov\_id" : "X109" ,

                "service\_prov\_name" : "Service Provider/3" ,

                "service\_prov\_type" : "non\_carrier" ,

        ...

        }

Connection. Current Working Assumption: interface will be connection-less.

The current CMIP interface is connection oriented (permanent between maintenance windows). The SOA/LSMS initiates a connection (called an “association”) to the NPAC. The NPAC never initiates a connection. Once a connection is established, requests/responses (i.e., messages) can be sent as long as the connection remains active, which is until the SOA/LSMS unbinds or the association is aborted. Today’s CMIP connections generally come up on Sunday morning, and remain up until the next weekend’s maintenance window starts.

In a connection-less environment (transient), each request establishes a connection (opens a port), sends the message, gets an acknowledgement, then tears down the connection (closes the port).

This also follows the paradigm of normal HTTP traffic flow. In addition to the client that makes requests, the local system would also implement a server to process responses. The request and the response (which constitute a single transaction) would be tied together with something like an invoke-ID or transaction ID.

Connection

Pro – NPAC needs to implement Server only, SOA/LSMS need to implement Client only, Service Providers do not need to open up a port in corporate firewall for NPAC-originated messages.

Con – more development since system needs to maintain state information, more development needed to handle potential of stale connections, need to implement heartbeat message to ensure connection is available, the more system resources are used the more system performance is impacted.

Connection-less

Pro – follows today’s paradigm of normal web traffic request processing, implementation does not need to maintain connection state information, connection persistence can be managed at the HTTP protocol level, idle/inactivity timouts can be managed at the network level, less use of system resources means better system performance.

Con – NPAC and SOA/LSMS need to implement both Client and Server, Service Providers need to open up a port in corporate firewall for NPAC-originated messages.

Session. Current Working Assumption: interface will be session-less.

In a session-based environment (e.g., online banking where you log in and validate your credentials), information is placed in your browser cache or a cookie (e.g., a key or token) that gets transmitted with every subsequent request for the duration of the session, such that you are validated from your initial login information, and do not need to re-validate each time. The server side maintains state information.

In a session-less environment, each request contains security validation that may be required for each new TCP connection attempt. There are options available for client authentication of each new connection request, such as a certificate that resides in the client (mandatory client certificate check as part of TLS handshake in order for the server to validate the requestor). Another option is a security key exchange on each request. This approach removes the need to link a period of time to the session between the SOA/LSMS and the NPAC.

Session

Pro – once session is authenticated upon login subsequent requests only require session validation.

Con – more development since session is persistent and needs to maintain state information on Server side, more system resources to maintain that persistence.

Session-less

Pro – less development since no need to maintain state information, flexible load-balancing can be used to manage workload (in HW and SW solutions) for both NPAC side and SOA/LSMS side in a web-services environment.

Con – requires system resources to perform security authentication on every request.

Push/Pull. Current Working Assumption: interface will push messages in real-time.

The current CMIP interface is push oriented. Whether originating from the SOA/LSMS or the NPAC, whenever a message needs to be sent, it is “pushed” out by the originator (Client, in CMIP called a Manager). In order for this to work in an HTTP environment, each side (participating end-point) needs to have both a Client (CMIP Manager) and a Server (CMIP Agent). Only the Client can initiate a request.

In a pull/poll environment, the SOA/LSMS will always be the Client, and the NPAC will always be the Server. The SOA/LSMS will periodically (e.g., every 5 seconds) ask the NPAC if there are any new transactions/messages for the local system. The implementation of pull/poll might be simpler (e.g., only a Client is required), however pull/poll will introduce unnecessary messages (may not be any new work to perform during that interval), and may require additional authentication for each pull/poll. The data synchronization will be dependent on the polling interval of the client.

Push

Pro – messages sent in real-time

Con – more complex development since NPAC and SOA/LSMS implement both Client and Server.

Pull

Pro – simpler development since NPAC implements Server only and SOA/LSMS implements Client only, message efficiency since sent in batch (e.g., if pull every 10 seconds, and 15 transactions generated during that interval, one message contains 15 transactions),

Con – messages not sent in real-time (e.g., if pull every 10 seconds, transaction could be 9 seconds “old”),

Interface Security. Current Working Assumption: end-point security will be achieved by enforcing HTTPS client-authentication during the connection establishment phase where NPAC generated keys are distributed to SOAs/LSMSs to facilitate HTTPS client-authentication.

The current CMIP interface uses a digital signature for each message/request. Once a secure association is established, messages are sent in binary encoded format.

In an HTTPS environment the message will be encrypted including some form of client authentication for each and every message.

Recovery. Current Working Assumption: recovery will be enhanced to deliver messages until successful.

The current CMIP interface uses a SWIM-based or time-based recovery method (SWIM = Send What I Missed). This requires the SOA/LSMS to request the recovery of missed messages. The queries related to recovery processing can be resource-intensive for both the NPAC and the SOA/LSMS.

Alternatively, in a “successful delivery” method, the NPAC would continue to send missed messages (tunable interval) until delivery is successful. This method would simplify the implementation and complexity of the system. Successful delivery would maintain a queue, and only send messages if the SOA/LSMS is accepting messages (existing NPAC functionality for Out-Bound Flow Control would limit the number of unresponded-to messages that have been sent). Another related option that can be considered is a message from the SOA/LSMS that indicates “I’m back online, go ahead and start sending now”.

SWIM Request/SWIM Response

Pro – SOA/LSMS recovery mechanism same as today’s CMIP.

Con – more development to implement recovery (request/response), recovery is resource intensive for both NPAC and SOA/LSMS, current messaging is queued until recover is completed.

Retry until Successful Delivery

Pro – no timeout or retry quantity to exhaust, only sends messages if SOA/LSMS capable of accepting message.

Con – retry continues even if SOA/LSMS not capable of successfully processing (can accept message but cannot process message).

**May ’12 APT**, discussion:

The group agreed to move forward based on the current working assumptions. If changes are needed during requirements development, it will be addressed at that time. In summary, here are working assumptions:

* Interface Protocol – HTTPS
* Interface Data Encoding – XML
* Interface Connection –connection-less
* Interface Session – session-less
* Interface Environment – push
* Interface Security – HTTPS client-authentication
* Missed Message Delivery – retry until successful

**Issue Tracking: (as of 6/30/12)**

Issue 1: Architecture Decisions

Status: Resolved

Description: Jan/Mar ’12 APT discussions documented in notes above. Need review, discussion, and decision on the following:

1. Interface Protocol, straight HTTPS (XML/JSON) or SOAP?
2. Data Encoding, XML or JSON?
3. Interface, Connection-oriented or Connection-less?
4. Interface, Session-oriented or Session-less?
5. Interface, Push or Pull?
6. Interface Security, Digital Signature or HTTPS message encryption/client authentication?
7. Recovery, SOA/LSMS initiated request or NPAC deliver until successful?

Refer to May ’12 discussion notes for resolution.

Issue 2: Security

Status: Resolved

Description: The security for the NPAC HTTP/XML interface needs to be discussed and consensus reached on how the interface will be secured. In recent discussions it was proposed to use HTTPS to provide encryption of all messages exchanged over this interface. For purposes of identification, a secure token or username/password approach was discussed. These discussions are aligned with the Neustar assumptions for security. Another option that could be discussed is embedding a digital signature similar to the existing CMIP interface. Other approaches could also be considered.

Refer to May ’12 discussion notes for resolution.

**FRS:**

TBD.

**IIS:**

Refer to the IIS, Part 2, CMIPversusXML – working draft (separate document).

**GDMO:**

None.

**ASN.1:**

None.

**XML:**

Refer to the XML schema – working draft (separate document).

**M&P:**

TBD.