\

NENA ALI Query Service Standard

NENA ALI Query Service Standard
NENA 04-005, Issue 1, November 21, 2006

Prepared by:
National Emergency Number Association (NENA) Technical Committee Chairs

Published by NENA
Printed in USA

NENA STANDARDS

NOTICE

The National Emergency Number Association (**NENA)** publishes this document as a guide for the designers and manufacturers of systems to utilize for the purpose of processing emergency calls. It is not intended to provide complete design specifications or to assure the quality of performance of such equipment.

NENA reserves the right to revise this NENA STANDARD for any reason including, but not limited to:

- conformity with criteria or standards promulgated by various agencies
- utilization of advances in the state of the technical arts
- or to reflect changes in the design of equipment or services described herein.

It is possible that certain advances in technology will precede these revisions. Therefore, this NENA STANDARD should not be the only source of information used. **NENA** recommends that readers contact their Telecommunications Carrier representative to ensure compatibility with the 9-1-1 network.

Patents may cover the specifications, techniques, or network interface/system characteristics disclosed herein. No license expressed or implied is hereby granted. This document shall not be construed as a suggestion to any manufacturer to modify or change any of its products, nor does this document represent any commitment by NENA or any affiliate thereof to purchase any product whether or not it provides the described characteristics.

This document has been prepared solely for the voluntary use of E9-1-1 Service System Providers, network interface and system vendors, participating telephone companies, etc.

By using this document, the user agrees that NENA will have no liability for any consequential, incidental, special, or punitive damages arising from use of the document.

NENA's Technical Committee has developed this document. Recommendations for change to this document may be submitted to:

National Emergency Number Association
4350 N Fairfax Dr, Suite 750
Arlington, VA 22203-1695
800-332-3911
or: techdoccomments@nena.org

Acknowledgments:

The National Emergency Number Association (NENA) CPE Technical Committee developed this document.

NENA recognizes the following industry experts and their companies for their contributions in development of this document.

Members:	**Company**
Mike Vislocky (Chair, CPE Committee)	Network Orange
David Hayes (Vice Chair, CPE Committee)	Verizon
John Sines (Working Group Leader)	HBF
Pierre Desjardins (Principal Document Author)	Positron
Guy Caron	Bell Canada
Martin Langlois	Bell Canada
Ben Lightner	BellSouth
Kevin Graham	HBF
Robert Sherry	Intrado
Gary Hutchins	Intrado
Martin Harnois	Plant Equipment
Kim Leigh	Qwest
Graham Chloupek	Siemens
Kevin Kleck	TC911
Barclay Hall	Verizon
Glenn Bowers	Zetron

TABLE OF CONTENTS

TABLE OF FIGURES

1 Executive Overview

1.1 Purpose and Scope of Document

This document defines the NENA XML ALI Query Service (AQS) that specifies new protocols between the PSAP and the Next Generation Emergency Services Network (NGESN). It provides the rationale behind the AQS and how it relates to the current ALI protocol. It also provides an overview of implementation alternatives (bindings) described in detail within the document.

1.2 Reason to Implement

This document describes specifications that overcome the following limitations of the existing ALI interface.

- Low-level character-based protocol supporting query request, query response and broadcast message concepts
- ALI data is limited to 511 characters
- The ALI data representation uses position-dependent encoding
- Mostly delivered over slow asynchronous links (as low as 1200 bps)

1.3 Benefits

Here are some of the advantages of AQS over the standing ALI protocol:

- Based on self-describing messages, defined by a formal XML syntax (schema)
  The advantage here is more expressive power and validation capability.
- Message exchange patterns are formally defined (WSDL[1])
  The advantage here is having an implementation (protocol) independent specification of the AQS interface (e.g. supporting Application Programming Interfaces)
- Each type of query is represented by a specific message definition
  The advantage here is the explicit and non-ambiguous statement of supported query requests
- The query key can support other forms beyond “NPA NXX TN”, such as SIP URL.

1.4 Operational Impacts Summary

As with any evolving technology, operations at the PSAP must adapt to manage the new evolving standard. Examples are:

- Most of these protocols assume TCP/IP connections to the PSAP.
- By delivering XML to the PSAP, the PSAP administrator, or a third party, must define how the XML data is represented to the Call Taker.

[1] Web Services Description Language

1.5 Document Terminology

The terms "shall ", "must" and "required" are used throughout this document to indicate required parameters and to differentiate from those parameters that are recommendations. Recommendations are identified by the words "desirable" or "preferably".

1.6 Reason for Issue

This document is issued to define the ALI Query Service that evolves the existing ALI interface. The purpose of this document is to recommend the minimal interfaces that may be used to evolve the ALI interface.

1.7 Reason for Reissue

NENA reserves the right to modify this document. Upon revision, the reason(s) will be provided in this paragraph.

1.8 Date Compliance

All systems that are associated with the 9-1-1 process shall be designed and engineered to ensure that no detrimental, or other noticeable impact of any kind, will occur as a result of a date/time change up to 30 years subsequent to the manufacture of the system. This shall include embedded application, computer based or any other type application.

To ensure true compliance, the manufacturer shall upon request, provide verifiable test results to an industry acceptable test plan such as Telcordia GR-2945 or equivalent.

1.9 Anticipated Timeline

It is expected that as part of the PSAP evolution that the protocols defined in this specification will be used to migrate from the position based ALI response to one using XML which may facilitate additional services.

1.10 Costs Factors

As with all new technology there is a cost to upgrade to new capabilities. However, it is assumed that the additional flexibility in migrating to an XML-based protocol will offset the initial expenditures to upgrade PSAP systems.

1.11 Cost Recovery Considerations

This document does not address cost recovery considerations.

1.12 Acronyms/Abbreviations

This is not a glossary! See NENA 01-002 - NENA Master Glossary of 9-1-1 Terminology located on the NENA web site for a complete listing of terms used in NENA documents.

The following Acronyms are used in this document:	
AQS	ALI Query Service
AQSI	ALI Query Services Interface
MEP	Message Exchange Pattern
NGESN	Next Generation Emergency Services Network
SC	Service Consumer
SP	Service Provider
WSDL	Web Services Description Language

1.13 Intellectual Property Rights Policy

1.13.1 General Policy Statement

NENA takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights.

NENA invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard.

Please address the information to:

National Emergency Number Association
4350 N Fairfax Dr, Suite 750
Arlington, VA 22203-1695
800-332-3911
or: techdoccomments@nena.org

2 Definition

The section provides the rationale behind the AQS and how it relates to the current ALI protocol. It also provides an overview of implementation alternatives (bindings) described in details in separate sections.

2.1 Current ALI Service

The current ALI query service can be summarized as follows:

- Low-level character-based protocol supporting query request, query response and broadcast message concepts
- ALI data is limited to 511 characters
- The ALI data representation uses position-dependent encoding
- Mostly delivered over slow asynchronous links (as low as 1200 bps)

The following diagram illustrates the syntax of the ALI request and response messages.

Figure 1 Legacy ALI Request/Response

The current philosophy is to issue redundant requests on dual-links to the ALI Service Provider (SP).

Refer to *NENA 04-005 Recommended Standards for E9-1-1 PSAP ALI Interface* (March 2005) for more details.

2.2 What About XML ALI?

The XML ALI representation is defined to deal with the following shortcomings associated with the current ALI data format; these are considered as limitations in the *Next Generation Emergency Services Network* (NGESN) context:

- static data representation; based on relative field positions and sizes (as opposed to a "*self-describing*" syntax such as one based on XML)
- lack of extensibility for accommodating new data elements
- requires a specific end-to-end agreement with each service provider

Now since XML is a textual representation of data, it could be thought of delivering XML ALI to the PSAP simply by having the current query response carry XML text instead of the usual ALI data text.

Figure 2 Simple XML Delivery

However, the same shortcomings cited earlier about the current ALI data format can also be raised about the current ALI protocol itself. It then becomes apparent that an XML-based protocol would better serve the delivery of XML ALI.

Such is the case for the *XML ALI Query Service*.

In any case, whatever the delivery method/protocol is used for getting XML ALI to the PSAP it inherits the following inescapable consequences:

- XML ALI payloads will often exceed the current 511 character limitation. This means a link speed upgrade is required in order to maintain equivalent ALI transmission times and in some cases may also require a CPE upgrade (old 911 controllers).
- XML ALI requires additional processing to parse the data and to "paint" the ALI display, since formatting information is not part of the data. That requires some form of PSAP software upgrade.

To provide some perspective on the subject of ALI transmission throughput, assume sending the ALI data on existing links. The table below compares the transmit times required for current ALI and XML ALI.

The reference ALI is a "full" ALI i.e. all fields filled with data. The size of the corresponding XML ALI (2124) *stream* includes the reference ALI data characters plus those required by the XML tags, but excludes white space characters.

Link Speed bps	**Bytes/sec**	**Reference ALI** transmit time 511 Bytes	**XML ALI** transmit time 2124 Bytes	Transmit time difference
1200	120	4.258	17.700	+13.4
2400	240	2.129	8.850	+6.7
4800	480	1.065	4.425	+3.4
9600	960	0.532	2.213	+1.7
19200	1920	0.266	1.106	+0.8
38400	3840	0.133	0.553	+0.4

Figure 3 ALI Transmit Times

The reference link speed used is 1200 bps since it is seemingly representative of many of the current ALI links. The table shows that 4.3 seconds are required to transmit the reference ALI at that speed.

To achieve matching throughput (quality of service) with the corresponding XML ALI, the table shows that a link speed of 9600 bps would be required (2.2 seconds). The 2 seconds advantage will later account for the XML data elements (currently defined or upcoming) that have no counterpart in the current ALI.

The **recommended minimum link speed is 19200 bps** to safely account for protocol overhead in the case of an AQS implementation using existing serial ALI links.

2.3 XML ALI Query Service

The XML ALI Query Service (AQS) defines the set of XML messages and message exchange patterns to be used in the delivery of XML ALI.

AQS messages and associated exchange patterns are described in detail in the Interface Specification section of this document in the form of an *abstract interface.* By definition an abstract interface is independent of any specific implementation (protocol stack) – representative implementations (also called *interface bindings*) are described in the Bindings section of this document.

Here are some of the advantages of AQS over the standing ALI protocol:

- Based on self-describing messages, defined by a formal XML syntax (schema)
  The advantage here is more expressive power and validation capability.
- Message exchange patterns are formally defined (WSDL)
  The advantage here is having an implementation (protocol) independent specification of the AQS interface (API)
- Each type of query is represented by a specific message definition
  The advantage here is the explicit and non-ambiguous statement of supported query requests
- The query key can support other forms beyond "NPA NXX TN", such as SIP URL.

2.4 How AQS Relates to the Current ALI Protocol

The following sample message diagrams will give an idea of how message parameters and payload, map from traditional ALI messages to AQS messages.

2.4.1 Request Message

A couple of concepts introduced by AQS are worth mentioning here: extensible query key and explicitly identified query properties.

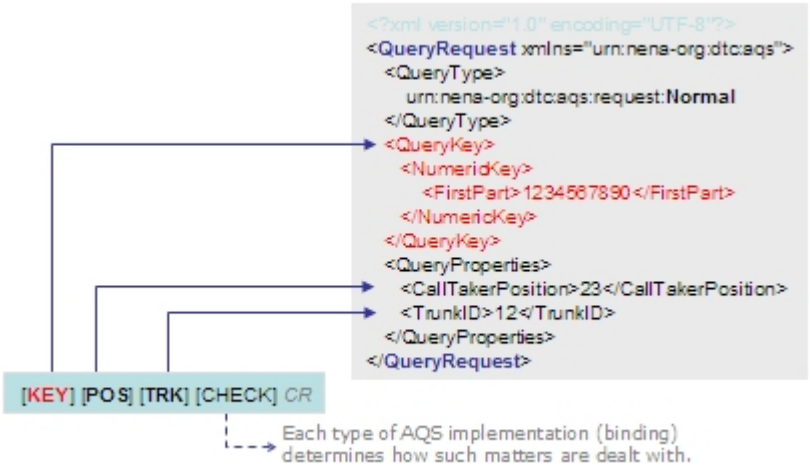

Figure 4 AQS Request Message

The *QueryKey* container element can easily be made to host different types of "keys" such as SIP URLs.

The *QueryProperties* container element is used to host complementary query parameters. This element is echoed (as is) in the corresponding response message.

As noted in Figure 4, each type of AQS implementations (binding) determines how transmission errors are detected to provide equivalent functionality of the checksum digit in the traditional ALI Query request.

2.4.2 Response Message

A couple of concepts introduced by AQS are worth mentioning here: explicit response status and opaque query result data type.

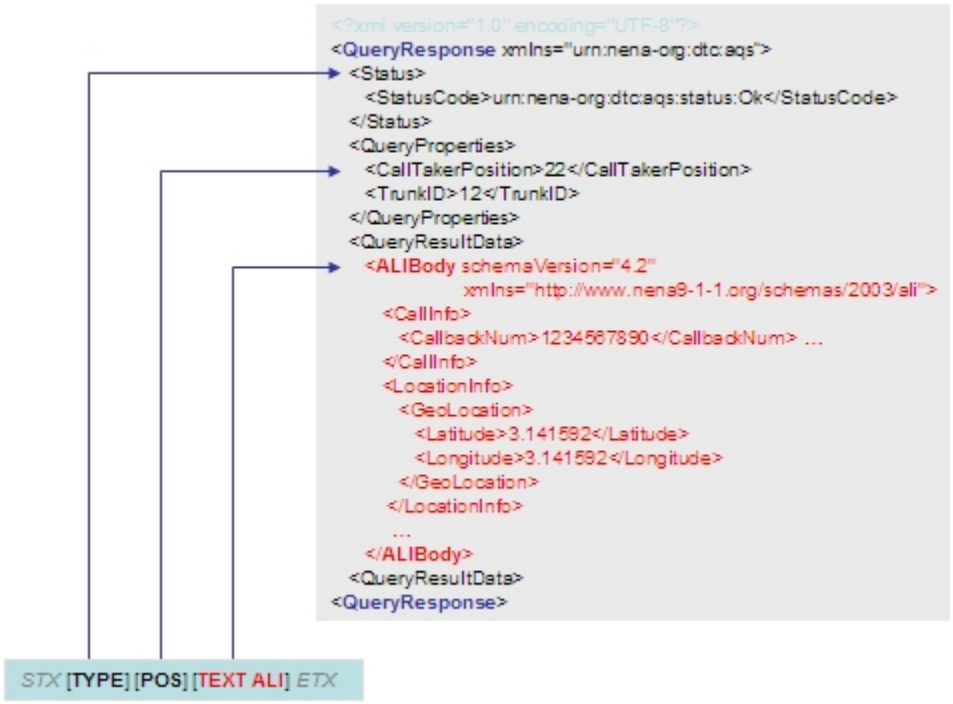

Figure 5 AQS Response Message

The *Status* container element is used to communicate response status information to any desired level of detail. Apart from the status code value, a corresponding textual status message can be included. An extensible status details element can also be included for application specific uses if required.

As currently defined, the *QueryResultData* container element restricts its content to be a standard NENA XML ALI, but could also be made to allow hosting other forms of XML data.

2.4.3 Broadcast Message

The current ALI protocol supports the concept of unsolicited notification through the broadcast messages: a "general purpose" (hex 35) message and a "going out of service" (hex 33) message. The syntax of these messages is similar to query response messages but they have no associated request messages.

AQS generalizes end extends this concept through the *Advisory* message. The sample message below illustrates how the "general purpose" broadcast message would be mapped and an AQS *Advisory*.

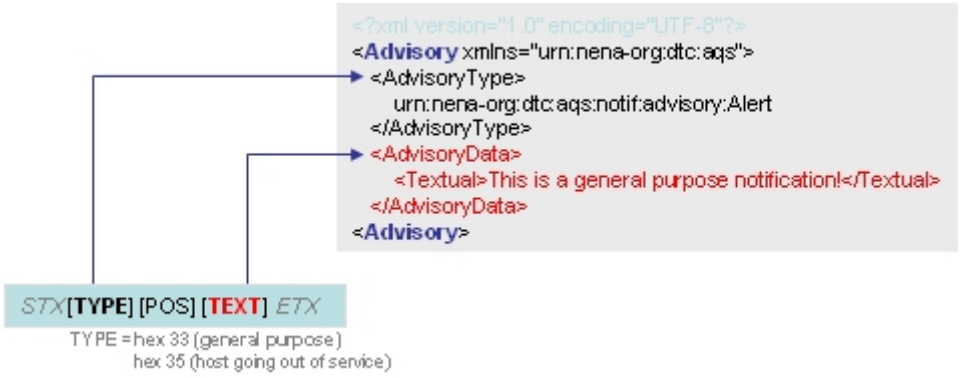

Figure 6 AQS Broadcast Message (Advisory)

2.5 AQS Implementations (Bindings)

An AQS binding defines a specific protocol (or protocol stack) that implements AQS abstract messages and interaction patterns.

In the case of AQS, a set of alternative bindings is being assessed. Alternatives have been classified in two categories:

- Bindings that reuse existing ALI links
- Bindings that use native-IP links

But why bother with existing ALI links when it is generally agreed that native-IP links will be the preferred communication channels used for PSAP-NGESN interactions? Because it is probable that there will be a need to have PSAP(s) benefit from XML ALI before native-IP links to NGESN(s) are generally deployed. Consequently, simple and low-cost interim bindings may be worth some consideration.

AQS bindings are described in detail in Section 26. Suffice it to say for now that the bindings under study are:

- Direct AQS messaging (reuses existing ALI links)
- Managed AQS messaging (reuses existing links)
- Direct web services (uses native-IP links)

- ESIF/TF34[2]-ESMI (uses native-IP links)

It is important to note that in all of the cases above, the processing application at the PSAP and ALI SP endpoints will always be handling AQS messages, as defined by the abstract interface, even though the binding differs. Ultimately, when the NGESN is ubiquitous, the interim binding(s) can then be discarded with relatively minimal change to the processing applications.

3 Interface Specification

3.1 Overview

This section is the technical specification of the AQS abstract interface (**AQSI**) i.e. a technology (implementation) agnostic description of the service.

The first part of the AQSI specification defines the XML messages that can be exchanged between AQS service consumers and providers. Message definitions are formally specified using an XML Schema.

The second part of the AQSI specification provides a description of the "application/business" message interactions that can occur between AQS service consumers and providers. The interactions are formally defined as message exchange patterns as a WSDL specification document.

In this document, the interface entities *service consumer (SC)* and *service provider (SP)* can be understood as representing respectively PSAP and ALI SP processing applications.

3.2 AQSI Messages

This section discusses XML schema characteristics and featured definitions for AQSI messages

3.2.1 Target Namespace

The AQS XML Schema types and elements are defined as belonging to the following namespace (also known as the *target namespace*):

```
urn:nena-org:dtc:aqs
```

This namespace name is also used as the basis for constructing other names used within AQS. For example, request status codes are expressed as values within that namespace. Code values then become contextualized values with respect to originating XML application (AQS) and ownership (in this case NENA Data Technical Committee).

The protocol schema (WSDL specification) of the NENA XML ALI Query Service defines a subordinate target namespace:

```
urn:nena-org:dtc:aqs:ws
```

WG comment (to be removed before publication)

[2] An ANSI standard under development by ATIS/ESIF Task Force 34 (ESMI: Emergency Services Messaging Interface and EISI: Emergency Information Services Interface)

The use of the URN syntax for namespace names is NOT consistent with the URL syntax used for the XML ALI Schema. By nature URN's support features (like uniqueness) that are important to the namespace name function.

NENA has not yet established a standard policy on namespace names. NENA may wish to study similar work ongoing within ESIF/TF34.

Note that the namespace identifier "nena-org" is a unit (the "-" is not an URN separator character).

3.2.2 Extensibility

Extensibility is one of the advantages of using XML, but is not automatic. Extension points need to be explicitly defined in the schema message definitions.

The AQS schema uses a dedicated element (**Extensions**) definition to achieve this. The content model for that element allows for any number of child elements as long as they are defined in a namespace other than the AQS namespace (the target namespace). Note that this element is defined such that the corresponding schema need NOT be available when validating the message.

This element is then referenced in message type definitions where extension points are desired (see the *tAbstractRequest* type for example).

3.2.3 Message Type Definitions

Two actors exchange messages in AQS: the *service provider* (an ALI SP for example) and the *service consumer* (a PSAP for example).

To guarantee consistency as well as facilitate use, consistency and evolution of schema message definitions, the AQS schema defines specific message types as derivations of base type definitions for request and response message classes.

These base types are marked as 'abstract' definitions in XML Schema terminology, which means that they cannot be used as the immediate type of an element. They can only be extended (or restricted) into specific types.

Each of the two base message types appear as underlined below and is followed by those specific message types that derive from it.

3.2.3.1 tAbstractRequest

This is the base type from which all AQS request message definitions are derived.

It defines:

- **version** (optional attribute): carries the version of the validating schema
  Version indicators are expressed as *Major.minor* pairs, where each pair member is a positive integer in the range 0..99 (a leading zero is insignificant).
- **ID** (optional attribute): carries a message identifier.

The value of this attribute should uniquely distinguish a message from all request messages exchanged between a given service consumer-provider pair. It is used for request-response message correlation (see the *inResponseTo* attribute of *tAbstractResponse* below).

- An optional extensibility point ("anyAttribute")

3.2.3.2 tQueryRequest

This type derives from *tAbstractRequest*. It adds the following features:

- **QueryType** (mandatory element): specifies the type of query to perform. The values for this element are URIs of the following form:
  - **`urn:nena-org:dtc:aqs:request:…`**

  Here are the currently defined query types :

  - **`urn:nena-org:dtc:aqs:request:Normal`**
    Initial ALI query
  - **`urn:nena-org:dtc:aqs:request:Rebid`**
    A repeated query made to retrieve the same (retransmit) or possibly refreshed XML ALI (possibly to get an updated caller location).
  - **`urn:nena-org:dtc:aqs:request:Refresh`**
    A repeated query made to retrieve a refreshed XML ALI (possibly to get an updated caller location). This query type can be used when a service provider can process repeated (retransmit) and refreshed queries differently – for example returning only updated XML ALI location data instead of a complete XML ALI.
  - **`urn:nena-org:dtc:aqs:request:Manual`**
    A discretionary (not associated with a 9-1-1 call) ALI query
  - **`urn:nena-org:dtc:aqs:request:Test`**
    A query used for testing end-to-end interoperability (PSAP-ALI SP)

  Application specific query types can be specified using the following URI syntax:

  - ***`urn:nena-org:dtc:aqs:request:appl-x:…`***
- **QueryKey** (mandatory element): accommodates different (exclusive) kinds of query keys:
  - **NumericKey** – is the traditional query key consisting of a mandatory **FirstPart** element and the optional element **SecondPart**
  - **URIKey** - a placeholder parameter (**URIKey**) is defined for future use in the context of VoIP networks
  - **Extension** (optional element), extensibility point
- **QueryProperties** (optional element): can accommodate ancillary information related to the query. The element content model includes:

- **TrunkID** (optional element), a trunk identifier (legacy parameter carried forward for backward compatibility)
- **CallTakerPosition** (optional element), a calltaker position index (legacy parameter carried forward for backward compatibility).
- **Extension** (optional element), extensibility point

3.2.3.3 tAbstractResponse

This is the base type from which all AQS response message definitions are derived.

It defines:

- **version** (optional attribute): carries the version of the validating schema
  Version indicators are expressed as *Major.minor* pairs, where each pair member is a positive integer in the range 0..99 (a leading zero is insignificant).
- **ID** (optional attribute): carries a message identifier.
  The value of this attribute should uniquely distinguish a message from all response messages exchanged between a given service consumer-provider pair.
- **inResponseTo** (optional attribute): references the associated request message. This identifier can be omitted if there is no corresponding request.
- An optional extensibility point ("anyAttribute")
- **Status** (mandatory element): carries result status information using the following child elements:

  StatusCode a mandatory element. The values for this element are defined by type *aqs:tStatusCode*, more specifically URIs of the following form:

  `urn:nena-org:dtc:aqs:status:……`

  The following status codes are defined :

  `urn:nena-org:dtc:aqs:status:Ok`

  This is the generic success status

  `urn:nena-org:dtc:aqs:status:OkMore`

  A service provider can use this status value in a *QueryResponse* message when it wishes to explicitly indicate to the service consumer that an updated or more precise XML ALI may be retrieved using a subsequent *rebid* or *refresh* query type request

  (ex: an initial response providing Phase I wireless location with Phase II location becoming available later).

  `urn:nena-org:dtc:aqs:status:OkPorting`

  Legacy ALI record porting in progress

  `urn:nena-org:dtc:aqs:status:OkMigrate`

  Legacy ALI record not unlocked

urn:nena-org:dtc:aqs:status:Requester

Generic status value for service consumer faults

(ex: QueryRequest message not well-formed or invalid)

urn:nena-org:dtc:aqs:status:Responder

Generic status value for service provider faults

(ex: out of resources, …)

urn:nena-org:dtc:aqs:status:VersionMismatch

The service provider does not support the AQS version (version attribute) specified in a QueryRequest message – the returned QueryResponse message should set its version indicator to that of the supported version.

urn:nena-org:dtc:aqs:status:RequestRefused

Service provider cannot honor the requested query

(ex: "manual" query requests)

urn:nena-org:dtc:aqs:status:DuplicateRequest

Service provider has discarded the request because it was considered redundant. Used specifically in reliability scenarios where duplicate requests are issued.

urn:nena-org:dtc:aqs:status:NotFound

No ALI corresponding to the query key specified in the QueryRequest was found

urn:nena-org:dtc:aqs:status:ANIFailure911

urn:nena-org:dtc:aqs:status:ANIFailure000

urn:nena-org:dtc:aqs:status:AnonymousCall

The status codes are categorized as follows (referring to the last member of each status code URI):

- Success codes - **Ok, OkMore, OkMigrate, OkPorting**
- Requester errors codes - **Requester, VersionMismatch, RequestDenied, DuplicateRequest**
- Responder errors codes - **Responder, NotFound, ANIFailure911, ANIFailure000, AnonymousCall**.

Application specific status codes can be specified using the following URI syntax:

urn:nena-org:dtc:aqs:status:appl-x:……

StatusMessage (optional): can carry textual status information

StatusDetails (optional): can carry any XML structured information

3.2.3.4 tQueryResponse

This type derives from *tAbstractResponse*. It adds the following features:

- An optional element (**QueryResultData**) used to carry query result data. Its content model is currently defined as a choice between the following:
  - A NENA XML ALI body (**ALIBody**)
  - Extensibility element (**Extension**)

  Note that this element is defined as optional so it may be omitted in cases of failed requests and no data is available.

- An optional element (**QueryKey**). This element is required in responses to failed requests. It should reproduce the exact contents of the corresponding query message element for the failed request. See **tQueryRequest** for content model.
- An optional element (**QueryProperties**). This element is required when the query message to which the response is associated includes such an element. It should reproduce the content of the latter without modifications. See **tQueryRequest** for content model.

3.2.3.5 tAbstractNotification

This is the base type from which all AQS notification message definitions are derived.

The tAbstractNotification message type includes:

- **version** (optional attribute): carries the version of the validating schema
  Version indicators are expressed as *Major.minor* pairs, where each pair member is a positive integer in the range 0..99 (a leading zero is insignificant).
- **ID** (optional attribute): carries a message identifier.
  The value of this attribute should uniquely distinguish a message from all response messages exchanged between a given service consumer-provider pair.
- **inRelationTo** (optional attribute): can carry the ID of a related query, response or notification message.
- An optional extensibility point (“anyAttribute”)
- **SenderID** (optional element): can be used to provide some identity of the sender.
  This attribute has no specific format (it is a simple string).
- **DateTimeIssued** (optional element): a timestamp of when the notification was issued.

3.2.3.6 tAdvisory

This type derives from *tAbstractNotification*. It adds the following features:

- **AdvisoryType** (mandatory element): used to distinguish types of advisories
  The values for this element are URIs of the following form:

  urn:nena-org:dtc:aqs:notif:advisory:…

  Here are the currently defined service provider advisories :

  urn:nena-org:dtc:aqs:notif:advisory:Alert

Generic advisory type for propagating different forms of alerts

(ex: Amber Alerts)

`urn:nena-org:dtc:aqs:notif:advisory:GoingOutOfService`

Used by the service provider to signal the service consumer that it wishes to terminate the service

`urn:nena-org:dtc:aqs:notif:advisory:Undetermined`

Identifies an advisory for which the type could not be determined because of a responder-side condition. Advisory data may not be provided in such cases.

Here are the currently defined service consumer advisories:

`urn:nena-org:dtc:aqs:notif:advisory:CallComplete`

Used by the service consumer to advise the service provider that the "call" associated with a previously issued *QueryRequest* is now complete. The *Advisory* message *inRelationTo* attribute is used to identify the associated request.Application specific extensions are accommodated by the following form:

`urn:nena-org:dtc:aqs:notif:advisory:appl-x:…`

- **AdvisoryData** (optional element): accommodates advisory type specific data
  The content model for this element accommodates and distinguishes textual as well as XML data using the following child elements (either or both can occur):
  - **Textual** can be used to carry advisory data as text
  - **Detailed** can accommodate any XML content belonging to a namespace other than the AQS namespace.

3.2.4 Query Request Message Definition

3.2.4.1 QueryRequest

This message (of type *tQueryRequest*) is used for all types of XML ALI queries. It is sent by a service consumer. The service provider responds with the *QueryResponse* message.

Here is a sample *QueryRequest* message of type "Normal":

```
<?xml version="1.0" encoding="UTF-8"?>
<QueryRequest xmlns="urn:nena-org:dtc:aqs"
xmlns:nali="http://www.nena9-1-1.org/schemas/2003/ali"
version="0.1" ID="XYZ123">
  <QueryType>urn:nena-org:dtc:aqs:request:Normal</QueryType>
  <QueryKey>
    <NumericKey>
       <FirstPart>1234567890</FirstPart>
    </NumericKey>
  </QueryKey>
  <QueryProperties>
    <TrunkID>23</TrunkID>
  </QueryProperties>
</QueryRequest>
```

Figure 7 - Sample *QueryRequest* message

By definition, the content of the *QueryProperties* element will be echoed without change in the corresponding *QueryResponse* message.

3.2.5 Query Response Message Definitions

3.2.5.1 QueryResponse

This message (of type *tQueryResponse*) is sent by a service provider in response to a *QueryRequest* message received from a service consumer. As seen later, a *QueryResponse* message can also be sent without having been solicited by a *QueryRequest* message (see Section 3.3.1.2).

Here is a response message corresponding to a successful *QueryRequest* request (in this example, the response message elements are *naqs* namespace qualified to clearly distinguish them from the default namespace XML ALI elements):

```
<?xml version="1.0" encoding="UTF-8"?>
<naqs:QueryResponse xmlns:naqs="urn:nena-org:dtc:aqs"
      xmlns:nali="http://www.nena9-1-1.org/schemas/2003/ali"
      inResponseTo="XYZ123" version="0.1" ID="ABC987">
       <naqs:Status>
              <naqs:StatusCode>urn:nena-org:dtc:aqs:status:Ok</naqs:StatusCode>
       </naqs:Status>
       <naqs:QueryProperties>
           <naqs:TrunkID>23</naqs:TrunkID>
       </naqs:QueryProperties>
       <naqs:QueryResultData>
              <ALIBody schemaVersion="4.2"
                       xmlns="http://www.nena9-1-1.org/schemas/2003/ali">
                     <CallInfo>
                            <CallbackNum>1234567890</CallbackNum>
                            <CallingPartyNum>0987654321</CallingPartyNum>
                            <ClassOfService code="1">String</ClassOfService>
                            <TypeOfService code="0">String</TypeOfService>
                            <SourceOfService>W</SourceOfService>
                     </CallInfo>
                     <LocationInfo>
                            <GeoLocation>
                                   <Latitude>3.141592</Latitude>
                                   <Longitude>3.141592</Longitude>
                            </GeoLocation>
                     </LocationInfo>
                     <Agencies>
                            <ESN>00000</ESN>
                     </Agencies>
              </ALIBody>
       </naqs:QueryResultData>
</naqs:QueryResponse>
```

Figure 8 - Sample *QueryResponse* message

Note that the QueryProperties of the related query (XYZ123) are reproduced intact.

If the query were unsuccessful, the *QueryResponse* message would look something like shown in the following illustration. In this case no *QueryResultData* is returned and the *QueryKey* is echoed from the corresponding request, as required when a request fails. The *QueryProperties* is echoed as in all cases where it is specified during the request.

```
<?xml version="1.0" encoding="UTF-8"?>
<QueryResponse xmlns="urn:nena-org:dtc:aqs"
      inResponseTo="XYZ123" version="0.1" ID="ABC987">
      <Status>
          <StatusCode>urn:nena-org:dtc:aqs:status:NotFound</StatusCode>
      </Status>
      <QueryKey>
          <NumericKey>
             <FirstPart>1234567890</FirstPart>
          </NumericKey>
      </QueryKey>
      <QueryProperties>
          <TrunkID>23</TrunkID>
      </QueryProperties>
</QueryResponse>
```

Figure 9 - Sample *QueryResponse* message for a failed *QueryRequest*

3.2.6 Notification Message Definitions

3.2.6.1 Advisory

This message can be issued either by a service provider or a service consumer.

Issuance of an Advisory message is independent of other messages, although it does support an optional attribute used for referencing some previously received or sent message (request, response or other advisory).

Here is a sample (this one is from a service provider):

```
<?xml version="1.0" encoding="UTF-8"?>
<Advisory xmlns="urn:nena-org:dtc:aqs"
      version="0.1" ID="ABC111">
   <AdvisoryType>urn:nena-org:dtc:aqs:notif:advisory:Alert</AdvisoryType>
   <AdvisoryData>
        <Textual>
           Amber alert advisory http://www.xyz.org/aa?id=123
        </Textual>
   </AdvisoryData>
```

Figure 10 - Sample *Advisory* message

3.3 AQSI Message Interactions

Previous sections discuss the different types of messages defined for the NENA XML ALI Query Service. This section describes the message exchange patterns (MEP) associated with the AQSI.

Web Service Description Language (WSDL) is used to formally express the MEPs. AQS can leverage the characteristic WSDL feature that allows describing the *abstract* (protocol agnostic) MEPs in a way that is relatively independent from the *concrete* protocol mapping (binding) descriptions.

This section is about an abstract definition for the AQS interface.

Bear in mind that the definitions below are abstract expressions of AQS message exchanges. They provide a useful context for understanding the role of the different message types defined in the AQS schema as well as how they are functionally related.

3.3.1 Message Exchange Patterns

WSDL terminology is service-centric, so in our case the *service provider* (or simply *service*) would be the ALI SP and the *service client* the PSAP. In protocol terminology, these are responder and requester entities respectively.

WSDL 1.1[3] defines the following types of message exchange patterns:

- **Request-Response**
  The service receives a message and sends back a response. An *operation* (service primitive) that uses such a pattern declares an *input element* followed by an *output element* (in that order). Use of a *fault element* is optional. Each element declaration is associated with a specific message type.
- **One-way**
  The service receives a message. The operation therefore has a single *input element*.
- **Solicit-Response**
  The service sends a message and receives a response. The operation therefore declares an *output element* followed by an *input element*. Use of a *fault element* is optional.
- **Notification**
  The service sends a message. The operation therefore has a single *output element*.

The AQS makes use of *Request-Response*, *Notification* and *One-way* patterns only.

Here are the AQS abstract service primitive (WSDL operation) definitions.

3.3.1.1 Request-Response MEP Service Primitives

Query service operation (primitive)

Service client sends message: *QueryRequest*

[3] WSDL 2.0 introduces additional MEPs

Service provider replies with message: *QueryResponse*

3.3.1.2 Notification MEP Service Primitives

Advisory service operation (primitive)

Service provider sends message: *Advisory*

PushResponse service operation (primitive)

Service provider sends message: *QueryResponse*

3.3.1.3 One-way MEP Service Primitives

Advisory service operation (primitive).

Service provider receives message: Advisory

This primitive establishes that the service consumer can also issue advisories.

4 Bindings

This section discusses the set of AQS bindings considered for assessment, as well as network configurations of interest to AQS.

4.1 Overview

In the Interface Specification section we introduced the abstract interface for AQS. Now look at implementation alternatives i.e. the method and protocols (also called *bindings*) by which the AQS messages get to and from the ALI SP.

Bindings have been classified in two categories:

- Bindings that reuse existing ALI links
  Direct AQS messaging (Section 4.4.1)
  Managed AQS messaging (Section 4.4.2)
- Bindings that use native-IP links
  Direct web services (Section 4.5.1)
  ESIF/TF34-ESMI (Section 4.5.2)

It is important to note that in all of the cases below, the processing application at the PSAP and ALI SP endpoints will always be handling AQS messages, as defined by the abstract interface, even though the binding differs. Ultimately, when the Next Generation Emergency Network (NGESN) is ubiquitous, the interim binding(s) can then be discarded with relatively minimal change to the processing applications.

4.2 Conformance to Abstract Interface

The AQS abstract interface section defines a number of service primitives (operations) with specific message exchange patterns.

The conformance requirements for any concrete mapping (binding) of the abstract service interface are as follows:

- A binding MUST support the *Query* service primitive
- A binding SHOULD support the *Advisory* one-way service primitive
- A binding MAY support all other service primitives, possibly using binding-specific MEPs

A binding may introduce alternative forms of non-mandatory operations as well as binding specific service primitives.

4.3 Representative Network Environments

Bindings are viable only if the network environment is appropriate. So before going into the bindings themselves, it would be worthwhile to highlight the network configurations that can host the AQS bindings under consideration.

It is recognized that there may be a number of variations to these configurations but the discussed representative configurations are sufficient to discuss the evolution topics as it relates to delivering XML ALI to the PSAP using AQS.

4.3.1 Legacy

Figure 11 shows the least sophisticated network configuration. The PSAP CPE connects directly to a pair of redundant ALI(s) using a private leased line configuration. These leased lines operate at a minimum of 1200 bits and have a realistic upper limit of 19.2 kilobits.

Figure 11 - Legacy network configuration

4.3.2 IP Backhaul

Figure 12 illustrates a step taken by a number of Emergency Services Providers (ESP) to extend the TCP/IP network. In this figure the same asynchronous connections from the PSAP CPE are at hand, however the CPE is connected to terminal server/router which converts the asynchronous connections to TCP/IP Telnet sessions. The ESP(s) TCP/IP network is used to back haul the request/response from the CPE to the appropriate ALI.

Figure 12 - IP backhaul network configuration

4.3.3 PPP/IP

Figure 13 illustrates a potential evolution of the *IP backhaul* configuration where the connection is end-to-end IP between the CPE and the ALI system. In this case the CPE could set up a TCP socket to each of the redundant ALIs. These connections would be persistent unless terminated via some abnormal actions. This configuration could also support HTTP bindings.

Figure 13 - PPP/IP network configuration

4.3.4 Native-IP/NGESN

Figure 14 shows a configuration where the PSAP CPE has a single native-IP connection to an advanced NGESN. In the context of NGESN, the PSAP CPE is viewed as a service consumer and the ALI(s) as service providers. Additional advanced services can also be made available in NGESN.

Notice that legacy one-to-one connectivity requiring dual links to redundant pairs of ALI(s) for each required ALI service is no longer necessary. The NGESN can take full advantage of dynamic connections between the PSAP CPE and services provided by the network. So depending on the AQS binding used, a service request may end up at ALI_a or any other ALI that runs an equivalent service.

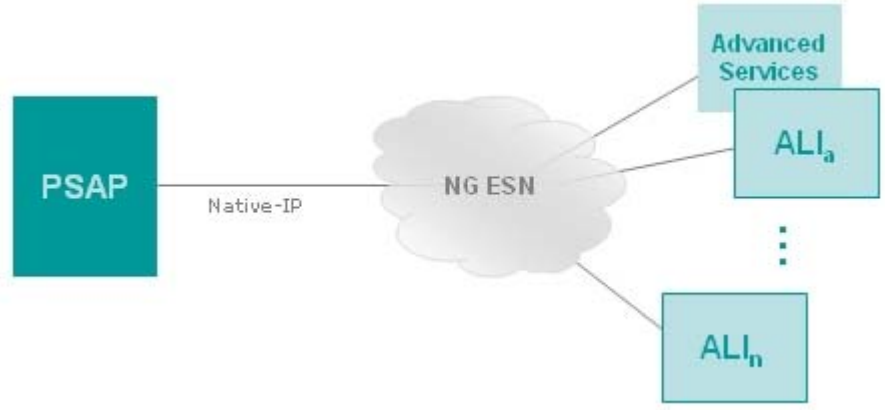

Figure 14 - Native-IP/NGESN network configuration

4.4 Bindings That Reuse Existing ALI Links

This section discusses AQS bindings that are essentially designed to work over existing (updated) ALI links:

- Direct AQS messaging (reuses existing ALI links)
- Managed AQS over PPP/IP/TCP (reuses existing links)

4.4.1 Direct AQS Messages

In this case, the traditional ALI request/response messages are replaced altogether with the corresponding AQS messages. The rest of the current ALI protocol conventions such as acknowledgements and timeouts remain as is.

The principal advantages are:

- incurs minimal change to infrastructure (link speed increase only)
- simple software modifications to application software

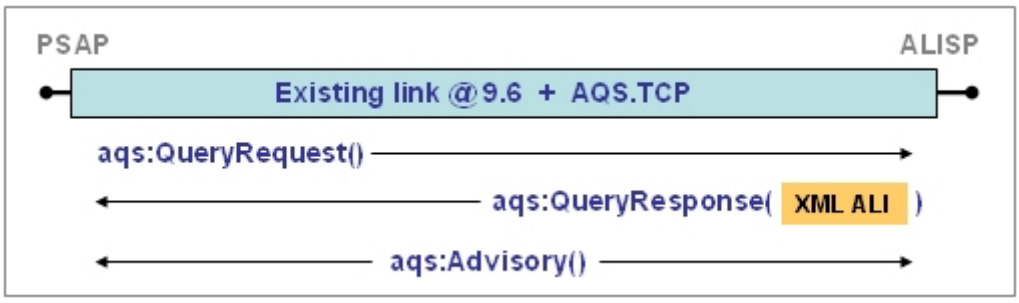

Figure 15 - *Direct AQS binding* illustration

In might make sense to systematically apply the STX/ETX framing (including the use of a BCC) used with the traditional ALI response message, to all the AQS messages.

It is worth mentioning that the CHECK and BCC bytes, as used in the traditional ALI request and response messages respectively, are in fact computed using the same XOR function applied to message bytes.

4.4.2 Managed AQS Messages

Here, PPP/IP communication channels are set up over existing links. Then an appropriate IP transport/application protocol is used for transiting AQS messages and session management.

The principal advantages here are:

- getting secure IP capability at low-cost and low complexity
- AQS message transmission integrity managed by IP transport/application protocols

TCP, HTTP or even SOAP could be considered as the transport/application protocol (in order of increasing complexity).

4.4.2.1 Using TCP

Here, the AQS query, query response and advisory messages are exchanged using TCP.

This binding is simple and thus easy to implement (read low cost). Because it rides on top of IP/PPP, compression (and security) features can be used if needed.

The binding uses a "session" concept to manage end-to-end state and keep interactions orderly. Additionally, because it is full-duplex, it naturally supports asynchronous messaging, offering direct support for AQS query interleaving and AQS advisory notification.

For this binding, the estimated link-speed requirement is 19.2 kbps, but 9.6 kbps could prove sufficient if the PPP compression feature is used. The 19.2 estimation is based on the following reasoning. As stated above, 9.6 is required just to account for the XML ALI payload size. Considering that PPP overhead is negligible, that TCP overhead is relatively small on quality links and that the messages defined by the TCP binding are small, doubling the links-speed requirement seemed more that enough.

A TCP-based binding called AQS.TCP and is illustrated in Figure 16 below. It is described in the AQS.TCP Binding section of this document. TCP naturally supports the asynchronous nature of the *aqs:Advisory* message, which would not be the case for HTTP or SOAP binding.

Figure 16 - *Managed AQS binding* (TCP version) illustration

4.4.2.2 Using HTTP

Here, the AQS query, query response and advisory messages are exchanged using HTTP interactions.

This binding requires adding HTTP server software on the ALI SP side.

In terms of link-speed upgrade requirements, this binding would probably also work with 19.2 (as for the TCP version).

4.4.2.3 Using SOAP

Here, the AQS query, query response and advisory messages are exchanged using SOAP/HTTP.

In view of the fact that AQS bindings that reuse existing ALI links is an interim solution to XML ALI delivery until NGESN becomes ubiquitous, there is no real advantage to use SOAP over the simplicity of the TCP version.

4.5 Bindings That Use New Native-IP Links

The following representative bindings work over a high-bandwidth native-IP link, most likely on a VPN (see Figure 14)

In any case, the usual network security issues now have to be considered.

4.5.1 Direct Web Services

This case is using an end-to-end SOAP implementation of AQS. The ALI service provider publishes a “processable” description (WSDL) of its service. The service consumer (PSAP CPE) uses that service description to access the service. This binding is described in detail in the AQS.SOAP Binding section (6) of this document.

This type of binding is used by ESIF/TF34-EISI, the service provider side of the TF34/ESNet.

Principal advantages:

- self-describing service interfaces
- numerous assistive development tools

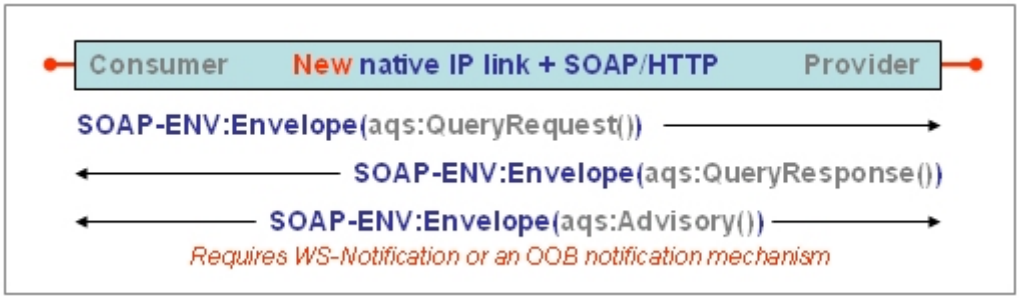

Figure 17 - *Direct web services AQS binding* illustration

4.5.2 ESIF/TF34-ESMI

Here the AQS messages are hosted within ESMI application messages (*esmi:*) used in the context of an Emergency Event dialog. As defined by ESMI, these messages remain independent of the actual request/response syntax of their payload (i.e. AQS). This binding is described in the AQS.ESMI Binding section (7) of this document.

Note that the *esmi:Notification* message is not considered part of an Emergency Event dialog. As for the other EMSI messages, it is independent of the actual syntax of its payload (i.e. the AQS Advisory message).

Principal advantages:

- inherent mediation capability
- advanced services beyond ALI provisioning

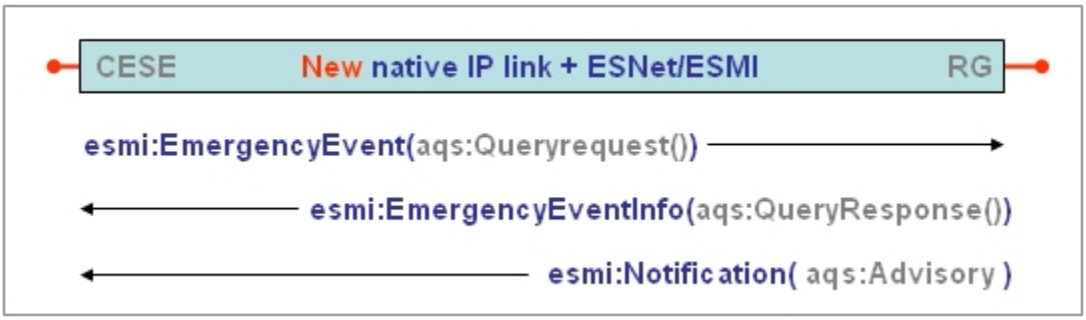

Figure 18 - *ESIF/TF34-ESMI binding* illustration

5 AQS.TCP Binding

This section provides a technical description of a sample binding for *Managed AQS messaging* based on TCP.

5.1 Introduction

The ALI Query Service (AQS) abstract interface specification defines the interactions and associated XML message formats for the provisioning XML ALI. The AQS interface is destined to replace the current legacy ALI character-based protocol. This document describes TCP-based *Managed AQS messaging* binding of the AQS interface (designed to be used over IP/PPP connections established over existing point-to-point ALI links).

Re-using existing point-to-point links for IP communications maintains the advantages relates to security issues, at the minimal cost of a link-speed upgrade (modem upgrade).

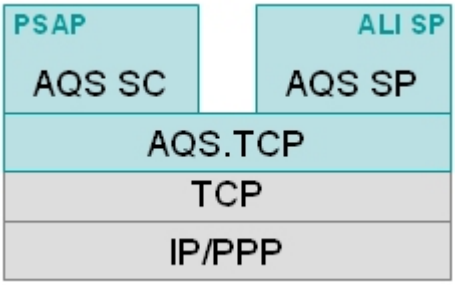

Figure 19 AQS TCP Protocol Stack

5.2 Namespaces

The AQS.TCP messages are defined to belong to the following namespace defined by the AQS.TCP schema (see Section 8):

urn:nena-org:dtc:aqstcp

In all AQS.TCP interactions, this namespace is assumed to be the default namespace i.e. the AQS.TCP message root elements should not be namespace-qualified.

The AQS.TCP namespace differs from the AQS namespace although the AQS.TCP schema does import AQS definitions. The AQS namespace is: ***urn:nena-org:dtc:aqs*** .

5.3 Binding

Referencing the AQS WSDL abstract interface definition document, the AQS abstract port type is mapped to a concrete TCP port type as follows:

- The *Query* operation, defined as an abstract request-reply MEP, is mapped (fragmented) into two distinct operations: a concrete one-way MEP operation called *Request* (carrying the *aqs:QueryRequest* message) and a concrete notification MEP operation called *Response* (carrying the *aqs:QueryResponse* message)
- The *PushResponse* operation, defined as a notification MEP, is directly mapped to a concrete notification MEP operation called *Response* (carrying the *aqs:QueryResponse* message).

- The *Advisory* (bi-directional) operation, defined as both an abstract notification and a one-way MEP, is directly mapped to concrete notification/one-way MEP operation also called *Advisory* (carrying the *aqs:Advisory* message)

Beyond the above, the AQS.TCP binding includes ancillary operations and associated messages used to support the concept of *service session* (session setup and teardown, session keep-alive and message receipt notification).

NOTE - There should be a 1:1 relationship between this binding and the communication link used. So in the case of dual ALI links, two individual instances of the binding would be used; one on each link. This is to minimize changes to the ALI SP side of the interface.

5.4 Service and Protocol Entities

This document refers to service and protocol entities using the following terminology:

- AQS (service) entities: **SC** (service consumer) and **SP** (service provider)
- AQS.TCP (protocol) entities: **requester** (client) and **responder** (server)

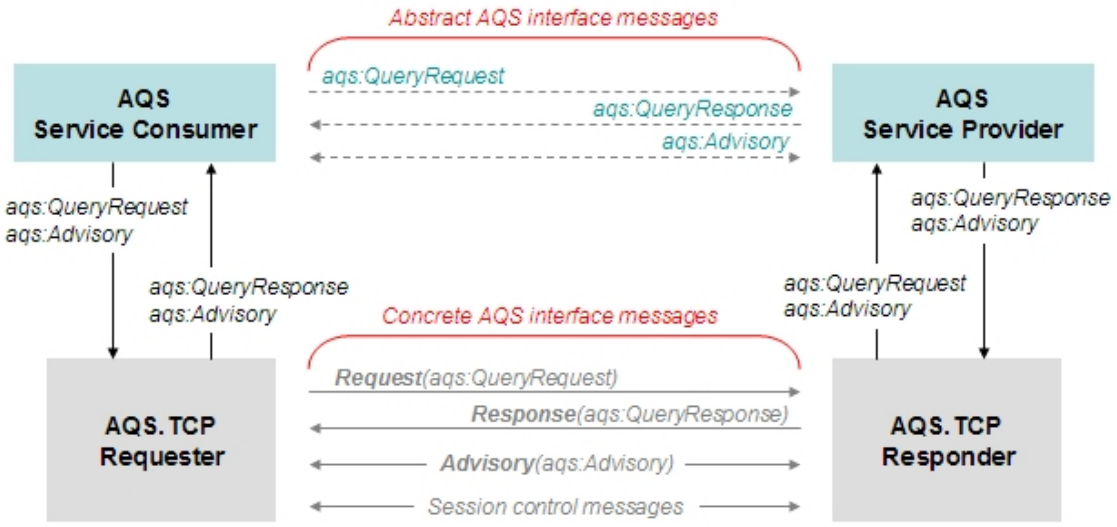

Figure 20 AQS Consumer and Provider Interactions

The above diagram represents the traditional use case. However, the binding is designed to also be applicable to the *pushed ALI* use case described in the appendix *Using AQS in "Pushed ALI" Scenarios*.

5.5 AQS Message Tunneling

AQS messages are not sent "in the clear" across an AQS.TCP session. Rather they are forwarded inside AQS.TCP specific wrapper messages. The wrapper messages are time stamped.

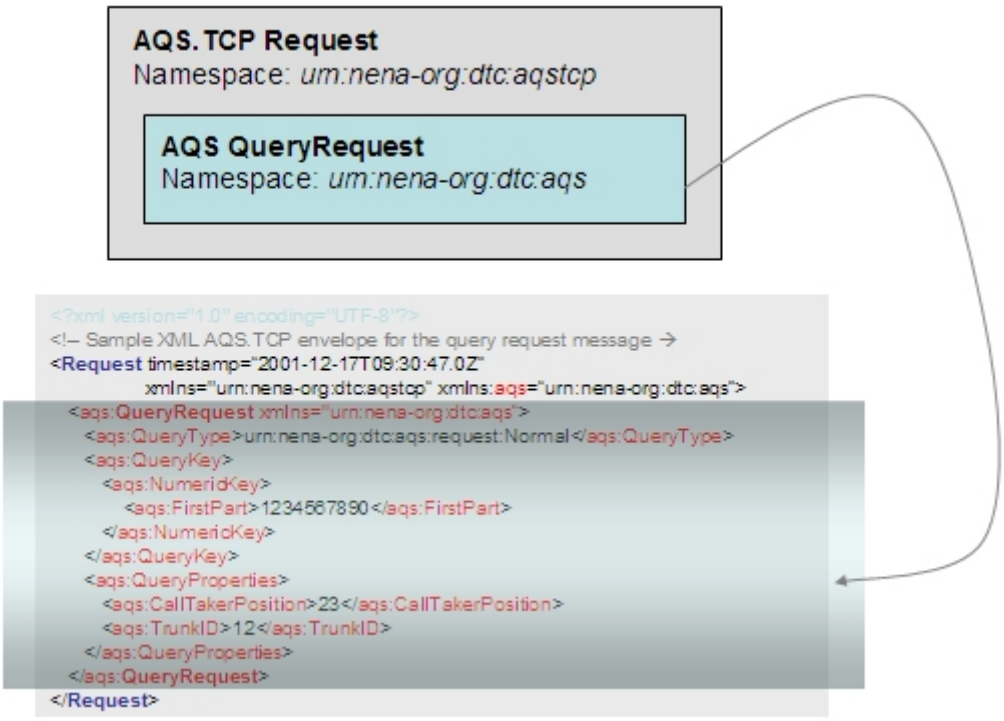

Figure 21 AQS TCP Tunneling

5.6 AQS.TCP Messages

In the following, AQS message references are namespace-qualified using the "***aqs***" prefix. References to AQS.TCP messages are not namespace-qualified since they are assumed to be in the default namespace i.e. the AQS.TCP namespace.

The AQS.TCP schema defining the messages is documented in the Exhibit section.

AQS.TCP messages fall in two categories described in detail below:

- Messages used for “session” control
- Messages used to tunnel the AQS messages

5.6.1 Session Control Messages

These messages are ancillary messages used for orderly management of end-to-end interactions between requester and responder entities.

The session may be established either by the requester or responder.

1. **SessionOpenRequest**

   This message is sent by the session initiator, to set up an AQS.TCP session. The normal response is a *SessionOpenResponse* message. Session setup is a synchronous sequence of interactions.

SessionOpenRequest/@version attribute is required for this message.

In cases where the responder initiates the session, the *SessionOpenRequest* message can optionally include elements that specify the *SessionOk* message behavior expected of the requester (these elements are defined under paragraph 5.*SessionOk* below).

2. **SessionOpenResponse**

   This message is sent in response to a *SessionOpenRequest*.

   The following *SessionOpenResponse/@status* values may be returned:

   - **`urn:nena-org:dtc:aqstcp:status:Ok`**

     Note that the session initiator will expect a *SessionOk* message before actually committing the session as open.

   - **`urn:nena-org:dtc:aqstcp:status:VersionMismatch`**
   - **`urn:nena-org:dtc:aqstcp:status:RedundantSession`**
   - **`urn:nena-org:dtc:aqstcp:status:ServiceNotReady`**

   In cases where the requester initiates the session, the *SessionOpenResponse* message can optionally include elements that specify the *SessionOk* message behavior expected of the requester (these elements are defined under paragraph 5. *SessionOk* below).

3. **SessionCloseRequest**

   This message is sent by a requester, as directed by a SC wanting to discontinue interacting with a SP, to teardown the current AQS.TCP session.

   Apart from local (SC) reasons, an SC should also trigger a session teardown when it receives a *urn:nena-org:dtc:aqs:notif:advisory:GoingOutOfService aqs:Advisory* message from the SP. This advisory indicates a SP's intention to stop processing queries. The SC should then refrain from issuing any new queries and trigger the requester to teardown the session only when it has no outstanding queries (if any).

   The responder is expected to respond with a *SessionCloseResponse* message.

4. **SessionCloseResponse**

   This message is sent by the responder in response to a *SessionCloseRequest*.

   The following *SessionCloseResponse/@status* values are supported:

   - **`urn:nena-org:dtc:aqstcp:status:Ok`**

   The requester is expected to confirm session teardown completion with an *Ok* message.

5. **SessionOk**

Periodically issued by the requester to implement a session "keep-alive" function when the session is "idle".

The responder is expected to confirm each *SessionOk* message with an *Ok* message.

It is always the requester entity that issues *SessionOk* messages, in accordance with the following parameters specified during session establishment:

SessionOkInterval is an optional parameter that can be used to set the time interval between *SessionOk* messages issued (by the requester entity) during session idle periods. This element is of type *xs:duration*.

SessionOkMissedLimit is an optional parameter that can be used to set the maximum number of *SessionOk* a responder entity will tolerate before declaring a session stale. This element is of type *xs:nonNegativeInteger*.

The above parameters are usually carried as optional elements of the *SessionOpenResponse* message. When the session initiator is the responder, it is the *SessionOpenRequest* message that may carry these parameters.

6. **Ok**

   This is a simple one-way message receipt acknowledgement. It can be issued by either the requester or responder.

   This message is allowed to carry implementation specific attributes i.e. from a namespace other than the AQS namespace.

5.6.2 AQS.TCP Tunneling Messages

These messages are the concrete interface messages used to transit (tunnel) AQS abstract messages between requester and responder end-points. References to status values below should not be confused with AQS-level status values (like QueryResponse/Status). Here the document is referring to binding-level status values defined in section 5.7

1. **Request**

   This message transits from requester to responder to forward an AQS query from the SC to the SP. The responder is expected to confirm reception with an *Ok* message.

   Message content is an integral *aqs:QueryRequest* message.

   Note that corresponding Response message delivery is considered to be an asynchronous interaction.

   No correlation between Request and Response messages is performed at the AQS.TCP level. Correlation of queries with query responses does however occur at the AQS level using AQS message identifiers.

2. **Response**

This message transits from responder to requester to forward an AQS query response from the SP to the SC. The requester is expected to confirm reception with an *Ok* message.

The *Response/@status* attribute is optional for this message.

3. **Advisory**

   This message transits from responder to requester to forward an unsolicited AQS advisory from the SP to the SC. The requester is expected to confirm reception with an *Ok* message.

   This message also transits from requester to responder to forward an unsolicited AQS advisory from the SC to the SP. The responder is expected to confirm reception with an *Ok* message.

   The *Advisory/@status* attribute is optional for this message.

   Message content is an integral *aqs:Advisory* message.

5.7 Status Values

Status indications are specified as URIs based on the AQS-TCP XML namespace. The following values are pre-defined:

`urn:nena-org:dtc:aqstcp:status:Ok`

This is the positive acknowledgement status.

`urn:nena-org:dtc:aqstcp:status:VersionMismatch`

The protocol version indicated by the session initiator, using the @version attribute, does not correspond to a supported version. The session initiator is expected to retry opening the session using one of the supported versions

`urn:nena-org:dtc:aqstcp:status:ServiceNotReady`

The service client (SP or SC) is not ready to process AQS requests. The session initiator is expected to retry establishing the session at a later time.

`urn:nena-org:dtc:aqstcp:status:RedundantSession`

A session is already been established. The session initiator is expected to respond with a *SessionOk* message and then proceed as it would when a session is established.

Application specific status values are accommodated using the following generic value syntax:

`urn:nena-org:dtc:aqstcp:status:appl-x:…`

5.8 Sequence Diagram

The following diagram only illustrates the general message flow for the usual use case where the requester initiates the AQS session. The diagram does not include handling of error conditions (such as timeouts).

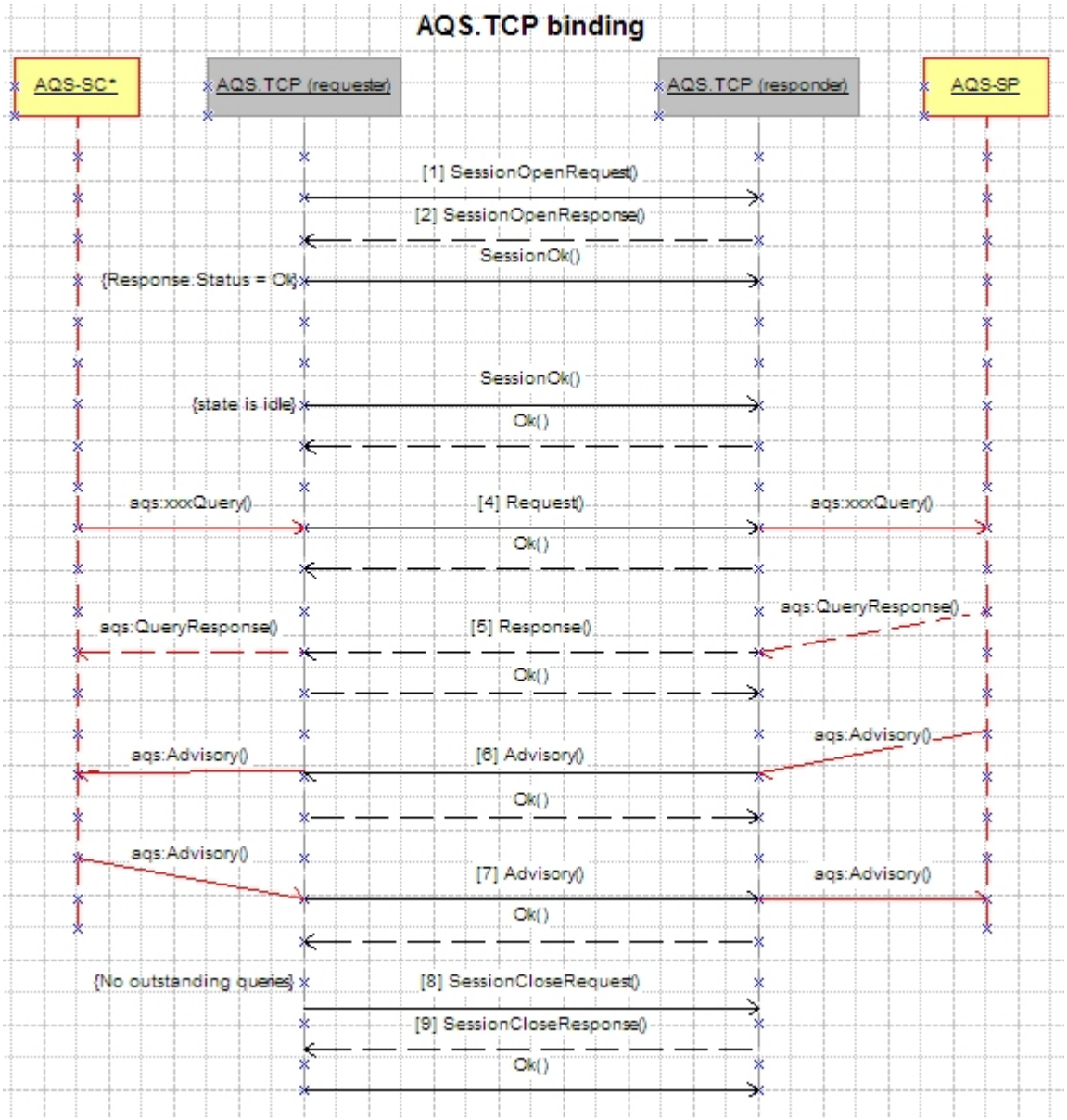

Figure 22 TCP Interaction Diagram

6 AQS.SOAP Binding

This section provides a technical description of a SOAP-based web service binding for *Managed AQS messaging* using.

6.1 Introduction

The ALI Query Service (AQS) abstract interface specification defines the interactions and associated XML message formats for the provisioning XML ALI. The AQS interface is destined to replace the current legacy ALI character-based protocol.

This document describes a web services SOAP-based *Managed AQS messaging* binding of the AQS interface (designed to be used over native IP links but could also be used over IP/PPP connections established over existing point-to-point ALI links).

The AQS.SOAP binding is a SOAP 1.2/HTTP binding[4].

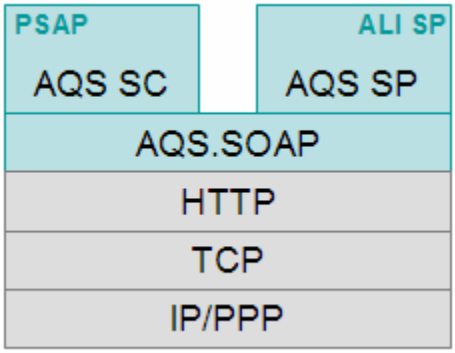

Figure 23 AQS SOAP Protocol Stack

6.2 Namespaces

The AQS.SOAP schema *includes* the definitions of the native AQS schema and introduces a couple of additional type and message definitions built from AQS definitions.

Consequently, the AQS.SOAP namespace is the same as the native AQS namespace.

The following table enumerates the namespace prefixes and associated namespace URIs used throughout this document:

Prefix	Namespace URI
aqs	**urn:nena-org:dtc:aqs**
soap	**http://schemas.xmlsoap.org/soap/envelope/**

[4] SOAP 1.2 Part 2 – Adjuncts (http://www.w3.org/TR/2003/REC-soap12-part2-20030624/)

6.3 Binding

Using the WSDL abstract interface definition (see Interface Specification) as reference, the AQS abstract port type is mapped to this concrete binding as follows:

- The *Query* operation, defined as an abstract request-reply MEP, is mapped as is to a SOAP in-out MEP
- The *Advisory* operation, defined as an abstract one-way MEP, is mapped as is to a SOAP in-only MEP
- The *Advisory* operation, defined as an abstract notification MEP, is alternatively mapped to the *GetAdvisory* operation defined as a SOAP in-out MEP (see Section 6.5.2). This MEP allows advisories to be acquired through explicit consumer requests rather than through unsolicited notifications (out-only).

  This does not preclude an implementation from providing a true notification MEP using more advanced web service techniques. A later edition of this binding may include a true notification MEP for the *Advisory* operation.
- The *PushResponse* operation, defined as an abstract notification MEP, has no counterpart in this binding.

  Creating an appropriate SOAP binding for a true notification MEP is not possible without using complementary interactions to control the notification context. This does not preclude an implementation from using advanced web service techniques (like WS-Notification) to extend this binding to include *PushResponse* support

The following diagram shows how AQS.SOAP messages are carried inside a SOAP envelope.

Figure 24 SOAP Envelope

6.4 AQS.SOAP Messages

This section defines message types and messages required by this binding that complement the AQS abstract interface definitions.

The following are formally defined in the AQS.SOAP schema (see Section 8).

6.4.1 *AdvisoryRequest* Message

This message (of type *aqs:tAdvisoryRequest*) is sent by the service consumer to retrieve a pending advisory (if any).

It is of type *aqs:tAdvisoryRequest*. The latter is a simple concrete instance of the *aqs:tAbstractRequest* virtual type which the AQS abstract interface uses as the base type for all request messages.

Here is a sample *AdvisoryRequest*:

```
<?xml version="1.0" encoding="UTF-8"?>
<AdvisoryRequest xmlns="urn:nena-org:dtc:aqs"
      version="0.1" ID="A123">
</AdvisoryRequest>
```

Figure 25 – Sample AQS.SOAP *AdvisoryRequest* message

6.4.2 *AdvisoryResponse* Message

This message is sent by a service provider in response to an *AdvisoryRequest* message, when one or more advisories are pending.

It is of type *aqs:tAdvisoryResponse*. The latter is an extended concrete instance of the *aqs:tAbstractResponse* virtual type which the AQS abstract interface uses as the base type for all response messages.

The content model is extended to include an optional *Advisory* element of type *aqs:tAdvisory* (the response payload). Because the *Advisory* element is wrapped by the *AdvisoryResponse* envelope, the *Advisory* attributes are rendered less relevant and will, in most cases, be absent

The *aqs:AdvisoryResponse/Status/StatusCode* element value will be one of the following success codes:

`urn:nena-org:dtc:aqs:status:Ok`

Indicates that a pending advisory was successfully retrieved

`urn:nena-org:dtc:aqs:status:OkMore`

Can be used by a service provider to indicate that one or more advisories are pending retrieval i.e. a subsequent request will return an advisory.

Here is a sample *AdvisoryResponse* for which successfully retrieved an advisory:

```
<?xml version="1.0" encoding="UTF-8"?>
<AdvisoryResponse xmlns="urn:nena-org:dtc:aqs"
      version="0.1">
   <Status>
      <StatusCode>urn:nena-org:dtc:aqs:status:Ok</StatusCode>
   </Status>
   <Advisory>
      <AdvisoryType>urn:nena-
org:dtc:aqs:notif:advisory:Alert</AdvisoryType>
      <AdvisoryData>
         <Textual>
            Amber alert advisory http://www.xyz.org/aa?id=123
         </Textual>
      </AdvisoryData>
   </Advisory>
</AdvisoryResponse>
```

Figure 26 – Sample AQS.SOAP *AdvisoryResponse* message

6.5 AQS.SOAP Interface Operations

This section formally defines AQS.SOAP operations and associated SOAP MEPs.

The AQS abstract interface was designed to be binding-agnostic. As such it defines a request-response message correlation mechanism built around native message identifiers carried as message attributes in request and response messages.

The inherent nature of SOAP request-response MEPs used by this binding makes message correlation implicit. AQS message identifiers may nevertheless be used for tracking, auditing or other purposes.

6.5.1 *aqs:Query* mandatory operation - in-out SOAP MEP

Service consumer sends message: *aqs:QueryRequest*
Service provider replies with message: *aqs:QueryResponse*
In all cases of AQS application error conditions, the service provider replies with a *soap:Fault* as defined below.

6.5.2 *aqs:GetAdvisory* optional operation – in-out SOAP MEP

Service consumer sends message: *aqs:AdvisoryRequest*
Service provider replies with message: *aqs:AdvisoryResponse*
For this operation, the interpretation assigned to the following status error code (*aqs:tStatusCode*) is that the service provider has no pending advisory to return to the service consumer:

`urn:nena-org:dtc:aqs:status:NotFound`

In all cases of AQS application error conditions, the service provider replies with a *soap:Fault* as defined below.

6.5.3 *aqs:Advisory* optional operation - one-way SOAP MEP

Service consumer sends message: *aqs:Advisory*

The WSDL description for the AQS.SOAP concrete binding can be found at the references provided in Section 8 (Exhibits).

6.6 AQS.SOAP Faults

The AQS abstract interface defines response status information, defined by type *aqs:tStatus*, as an integral part of all response messages. Status codes, defined by type *aqs:tStatusCode*, are defined to represents success as well as failure conditions.

This binding requires that all AQS error conditions will be propagated back to a service consumer by using the SOAP fault mechanism instead of using the status information embedded in the response message.

SOAP 1.2[5] requires that for indicating faults, the *soap:Fault* element be the only element in the *soap:Body*.

The AQS.SOAP binding specifies usage conventions only for the following *soap:Fault* sub-elements when reporting any AQS application specific error condition:

- ***Soap:Fault/Code/Value*** (mandatory)

  soap:Sender

  Specify for all AQS requester error codes

  soap:Receiver

  Specify for all AQS responder error codes

  AQS requester and responder error codes are defined under type *aqs:tAbstractResponse* (*StatusCode* element) as part of the Interface Specification.

- ***soap:Fault/Code/Subcode/Value*** (mandatory)

  The assigned values should map the AQS requester and responder error codes defined by the *aqs:tStatusCode* type as follows:

  aqs:xxx, where "*xxx*" is the last member of a status code URI defined by *aqs:tStatusCode*.

  For example, assume the following status code URI:

  `urn:nena-org:dtc:aqs:status:NotFound`

  The above status code URI would map to a fault subcode value of

[5] The SOAP 1.1 definition differs here.

aqs:NotFound

Application specific error status codes i.e. those *aqs:tStatusCode* URIs that have the last token member preceded by the "appl-x" token, map to a "dash" separated concatenation of the last two tokens. For example:

`urn:nena-org:dtc:aqs:status:appl-x:FooCode`

would map to a fault subcode value of

`aqs:appl-x-FooCode`

- ***soap:Fault/Reason*** (mandatory)

  This element is made mandatory by SOAP 1.2. However, the AQS.SOAP binding does not impose pre-defined text for this element.

- ***soap:Fault/Detail*** (optional)

  May be used to include contextual information related to the fault being reported. One or more of the following elements may be present.

  aqs:Status (optional)

  Must be present for AQS specific faults.

  If a value is supplied, it should be of type *aqs:tStatus*

  The value of *aqs:Status/aqs:StatusCode* conveys the AQS native status error code that corresponds to the fault. It should be semantically equivalent to the value specified for Fault/Code/Subcode/Value.

  aqs:RelatedTo (optional)

  Conveys the identifier of the AQS message received from the requester that resulted in the fault. Must be present if the requester message included such an identifier.

  If a value is supplied, it should be of type *aqs:tTransactionID*.

  aqs:ExpectedVersion (optional)

  Conveys the AQS version supported by the receiver, when a aqs:VersionMismatch fault is reported.

  If a value is supplied, it should be of type *aqs:tSchemaVersion*.

  aqs:QueryKey (optional)

  Must be present for failed *aqs:QueryRequest* operations. Replicates the *aqs:QueryKey* used with the failed request.

  If a value is supplied, it should be of type *aqs:tQueryKey*

  aqs:QueryProperties (optional)

Must be present for failed *aqs:QueryRequest* operations that included such an element. Replicates the *aqs:QueryProperties* in the failed request.

If a value is supplied, it should be of type *aqs:tQueryProperties*

6.7 Service Reliability Using Duplicate Queries

There are situations where it will be a requirement to issue duplicate queries to address reliability concerns.

Whether this is achieved by issuing the queries over two different consumer-side network connections or through one consumer-side network connection but targeted at redundant service port instances, the service provider needs a way by which it can distinguish those queries that are redundant, i.e. have already been processed.

As stated earlier, the inherent nature of SOAP request-response MEP used for the Query operation makes message correlation implicit. AQS message identifiers may nevertheless be used for tracking, auditing or similar purposes where there is a need to distinguish transactions.

Achieving service reliability using duplicate queries SHALL be as follows:

- the service consumer performs duplicate *aqs:Query* operations by using the same message ID value in two *aqs:QueryRequest* messages.
- the service provider uses ID attribute value equality as the basis for discarding redundant queries.

  As used here, the identifiers assigned to the ID attribute need to be globally unique i.e. a service provider must be able to distinguish queries from different requesters as well as from a given requester[6].

- because *asq:Query* operations follow a SOAP request-reply MEP, the service provider must send back a response even for those queries deemed redundant. To this end, the service provider shall return a "DuplicateRequest" SOAP *Fault* as shown below.

  Note that the *aqs:RelatedTo* element in *soap:Fault/soap:Detail* should be set to the ID attribute value associated with the discarded query.

Figure 27 illustrates a SOAP Fault message corresponding to the *aqs:QueryRequest* example of Figure 7 – notice the replication of *aqs:QueryProperties* and *aqs:QueryKey* elements as required in section 6.6.

[6] There is no notion of "session" in SOAP, only individual exchanges of request-response message pairs.

```
<soap:Envelope
  xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/
  xmlns:aqs="urn:nena-org:dtc:aqs">
   <soap:Body>
     <soap:Fault>
         <soap:Code>
            <soap:Value>soap:Requester</soap:Value>
            <soap:Subcode>
               <soap:Value>aqs:DuplicateRequest</soap:Value>
            </soap:Subcode>
         </soap:Code>
         <soap:Reason xml:lang=”en”>AQS Application fault</soap:Reason>
         <soap:Detail>
             <aqs:RelatedTo>XYZ123</aqs:RelatedTo >
             <Status>
               <StatusCode>urn:nena-org:dtc:aqs:status:DuplicateRequest
               </StatusCode>
             </Status>
             <QueryProperties>
               <TrunkID>23</TrunkID>
             </QueryProperties>
             <QueryKey>
                <NumericKey>
                   <FirstPart>1234567890</FirstPart>
                </NumericKey>
             </QueryKey>
         </soap:Detail>
     </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

Figure 27 – Sample AQS SOAP *Fault* message

7 AQS.ESMI Binding

This section provides a technical description of a sample binding for *Native IP AQS messaging* based on ESIF's Emergency Services Messaging Interface (ESMI).

7.1 Introduction

The ALI Query Service (AQS) abstract interface specification defines the interactions and associated XML message formats for the provisioning XML ALI. The AQS interface is destined to replace the current legacy ALI character-based protocol. This document describes ESMI-based *Native IP AQS messaging* binding of the AQS interface.

Taking advantage of native IP capabilities allows the PSAP not only to replace existing ALI query/response but subscribe to new services. ESMI provides a bi-directional communications link between the PSAP and the Emergency Services Network (ESNet).

The following figure illustrates the communication protocol stack supported by the ESMI. The lower layers of the protocol are standard IP layers. The Emergency Services Transfer Protocol (ESTP) is the wrapper for the application business logic messages, called ESMI Application Messages (EAM). SIP is used for connection management between the PSAP entity (the Conforming Emergency Services Entity [CESE]) and the Emergency Services Network's Response Gateway (RG).

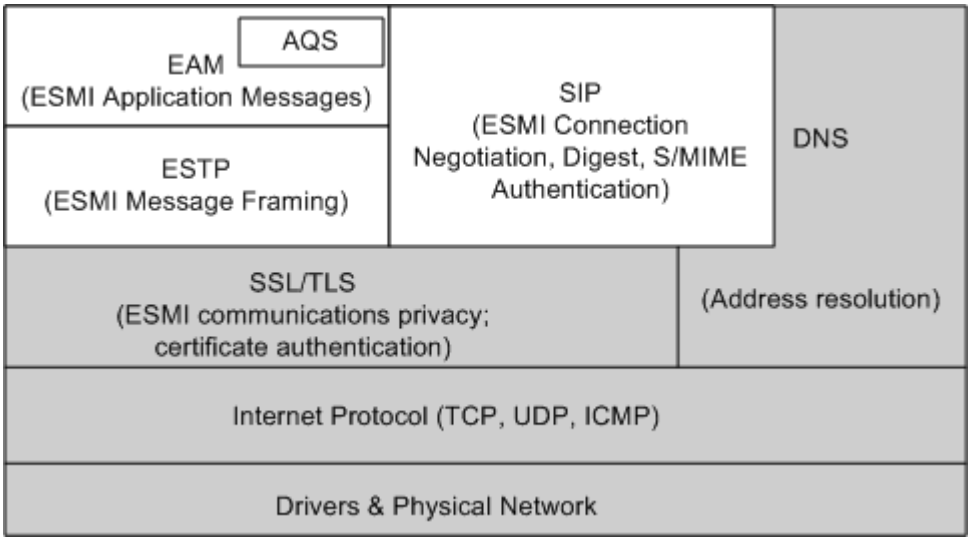

Figure 28 ESMI Protocol Stack

7.2 Namespaces

The ESMI uses messages that belong to the following namespace:

```
urn:atis-org:esif:xml:ns:esi:esmi
```

For the message bodies (payloads) it is assumed that the NENA AQS namespace will apply as defined section 3.2.1.

7.3 Binding

In ESMI the AQS bodies are indirectly bound to ESTP since they are embedded within an ESMI message to the Emergency Services Transfer Protocol (ESTP). ESTP provides a bi-directional wrapper for the message bodies.

7.4 Service and protocol entities

This document refers to protocol entities using the following terminology:

- Conforming Emergency Services Entity (CESE) – Entity at the PSAP requesting services
- Response Gateway (RG) – Entity in the ESNet mediating services for the CESE.

7.5 Wrapping AQS messages

A multi-layered approach to the protocol suite is used to separate the set of business messages from the underlying utility functions of message packaging and transmission. This allows the introduction of new messages to the ESMI standard over time. Connection negotiation is also separate from the messaging protocol itself to allow for intelligent connection establishment.

Session Initiation Protocol (SIP) is used to set up the application session over which ESTP transmits ESMI application messages. The SIP messages within ESMI are meant to negotiate parameters between the two end points for the forthcoming application session. Essentially the session negotiation will encompass the authentication and authorization of both parties and the negotiation of a media channel over which the ESMI Application Messages will pass.

AQS messages are encapsulated by the EAM application message construct within the context of the ESTP message wrapper.

Emergency Service Transfer Protocol (ESTP) is an application protocol which supports the bi-directional transportation of Emergency Services Messaging Interface application messages. Because TCP is a stream oriented protocol, it has no built-in constructs to delineate a "message" within the stream. ESTP provides this delineation.

ESMI Application Messages (EAMs) are the collection of application messages that are currently defined by ESMI. These messages carry business data and express business interactions between a CESE and the RG. ASQ messages constitute the body of the EAM.

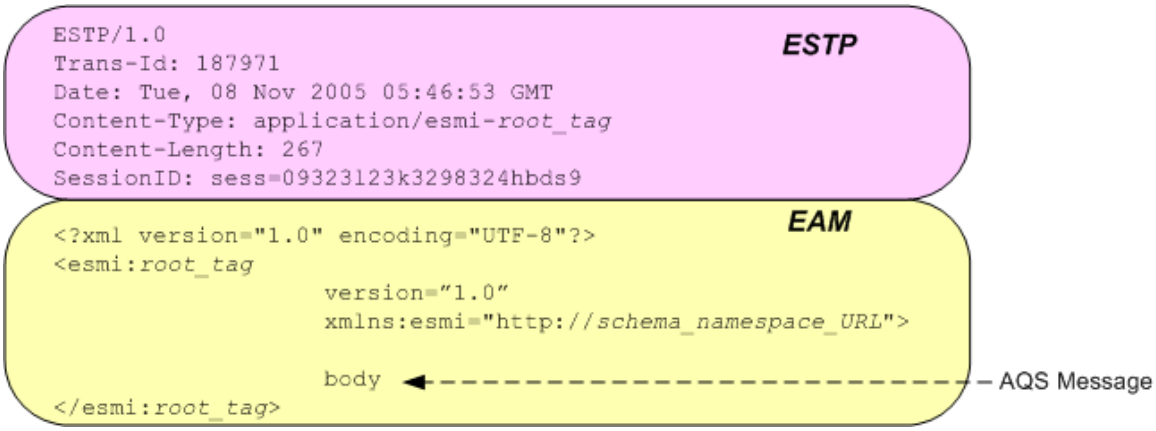

Figure 29 ESMI Messaging

7.6 AQS.ESMI Messages

This section outlines the messages defined in ESMI that are use to implement the ALI Query Service. The SIP messages are the standard connection management messages. The EAM messages carry the body of the AQS XML elements.

AQS.ESMI messages fall in two categories described in detail below:

- Messages used for application session control (SIP)
- Messages used for business logic (EAM)

7.6.1 Session Control Messages Using SIP

The standard SIP session negotiation is shown in the figure below. The CESE requests the connection and the RG authenticates. The application connection is negotiated and set up. Then the CESE and RG begin to exchange ESMI Application Messages. At some point the CESE may want to terminate the application connection, e.g. to acquiesce to another application connection.

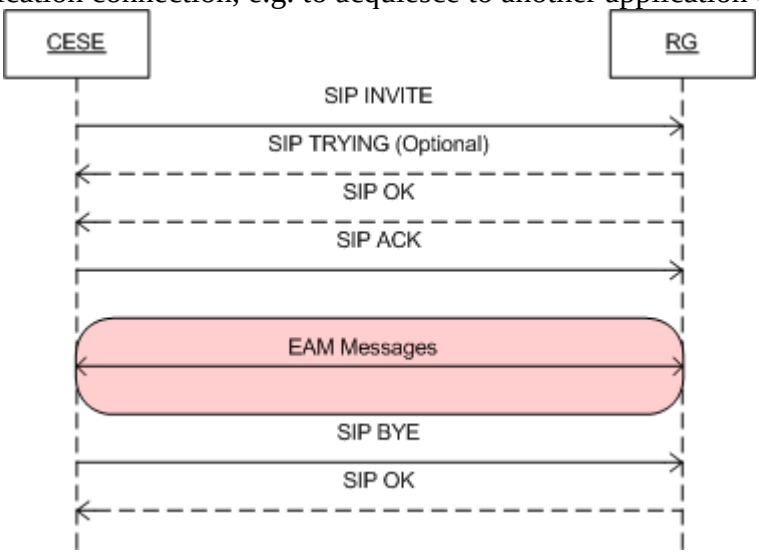

Figure 30 ESMI Session Control Sequence Diagram

7.6.2 AQS.ESMI Application Messages

The messages defined in this section are only those that apply to the ALI Query Service. Other messages are defined in the ESMI specification and the reader is encouraged to review that document.

1. **Emergency Event Message**

   This is the simplest and in some ways the archetypal message flow between the RG and the CESE. Most of the service message sequences follow the pattern that the CESE requests data from the RG using a Key (e.g., TN or ANI, ESRD, ESRK, MDN, etc.), receives location information retrieved with that key from the RG, and then sends an Emergency Event completion message to the RG. Depending upon the service and information contained within the ESNet multiple EEInfo messages may be sent to the CESE. The payload of this message is an AQS *QueryRequest* message with a query type of "*urn:nena-org:dtc:aqs:request:Normal*".

2. **Emergency Event Information Message**

   An Emergency Event Information message is sent from the RG to the corresponding CESE to carry information related to a given service invoked as a result of an Emergency Event message. There is the potential that some Emergency Events will result in multiple responses from the RG to the CESE for a single Emergency Event. The payload of this message is an AQS *QueryResponse* message

3. **Emergency Event Update**

   During the context of processing an Emergency Event there may be a need for a CESE to request an update of the service related information received in previous EEInfo Messages. An example of this is location information related to a Wireless Phase 2 call. The CESE does this by sending an EEUpdate message to the RG. This message contains the specific service for which the update is being requested. The payload of this message is an AQS *QueryRequest* message with a query type of "*urn:nena-org:dtc:aqs:request:Rebid*".

7.7 Sequence Diagram

The following diagram only illustrates the general message flow but does not, for example, cover handling of error conditions (such as timeouts). Some EAM messages use the ALI Query Service messages within the body of the message (e.g. EEvent and EEInfo). Other messages (e.g. EEventResp) provide capabilities beyond the scope of the ALI Query Service.

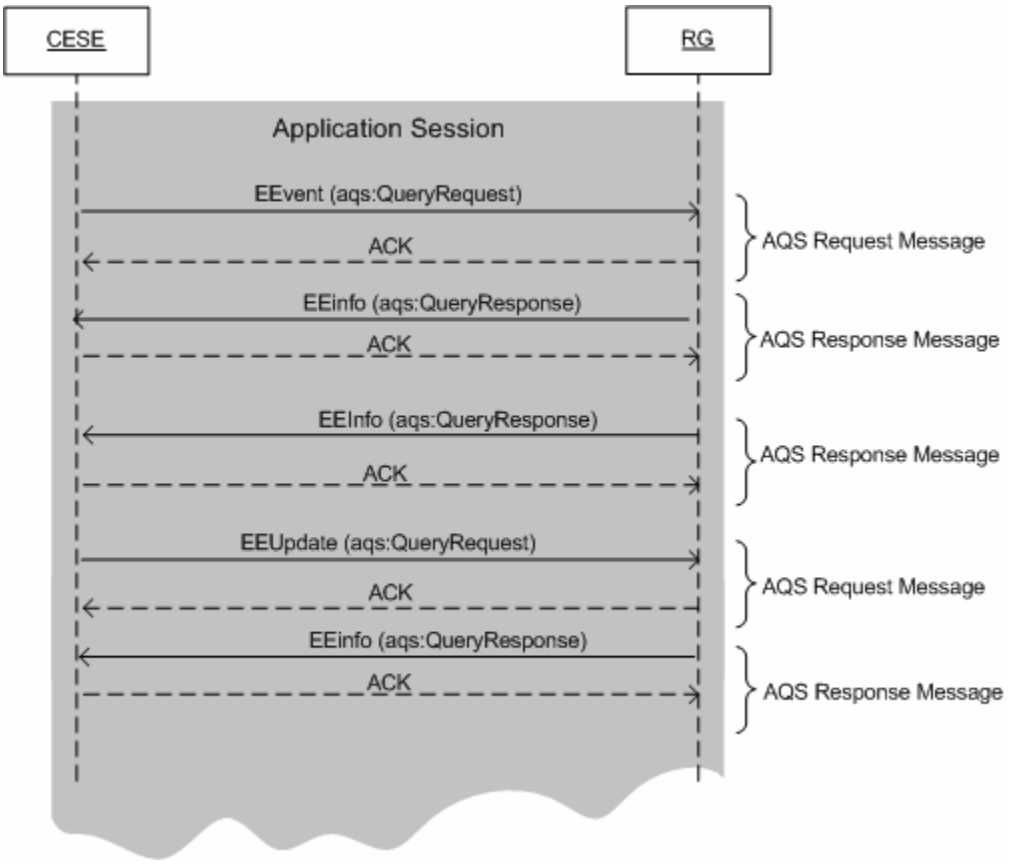

Figure 31 ESMI Application Sequence Diagram

The following diagram shows the Notification message being sent from the RG to the CESE to inform the CESE of an incident of interest.

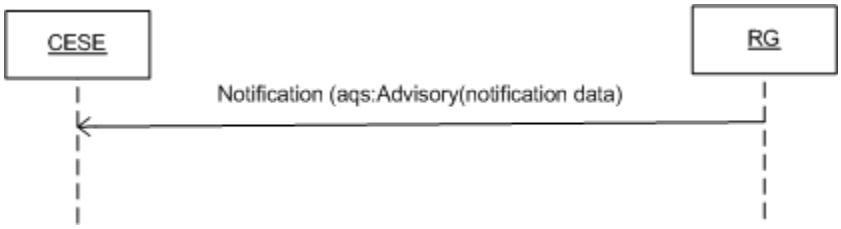

Figure 32 ESMI Notification Sequence Diagram

8 References

9 Exhibits

The following are the list of XML schema and WSDL documents that support this standard. The documents may be found on the NENA web site at the following address “hyperlink”.

WG Comment – Deal with hyperlink issue here (to be removed at publication)

- AQS.I – Generic WSDL
- AQS.ESMI.I – ESMI WSDL
- AQS.SOAP.I – SOAP WDSL
- AQS.TCP.I – TCP WSDL
- AQS.SOAP-schemadoc (HTML schema documentation)

AQS.TCP-schemadoc (HTML schema documentation)

Appendix –Using AQS in "Pushed ALI" Scenarios

This Appendix serves as a standardized application note. It describes how a pushed ALI scenario SHOULD be implemented in an AQS context (currently limited to the AQS.TCP binding).

In "pushed ALI" scenarios, the service consumer does not issue query requests to get ALI; the service provider performs unsolicited "pushes" of ALI to the consumer.

This is typical of Centrex-like Emergency Services Network interfaces, where the ALI query is performed on the network side.

Note - This Appendix is structured in main sections that correspond to those found in the AQS documents for which additional information is required.

A.0 Interface Specification – additional information

This section provides additional information for the use of the interface specification in "pushed ALI" scenarios.

A.0.1 QueryRequest Message Definition

QueryRequest messages are not used in pushed-ALI scenarios.

A.0.2 QueryResponse Message Definition

The *QueryResponse/@inResponseTo* attribute, that is normally used to correlate request and response messages, SHOULD NOT be present in pushed messages.

A.0.3 PushResponse Service Primitive

The abstract *PushResponse* notification service primitive SHOULD be used to push *QueryResponse* messages to the service consumer.

A.1 AQS.TCP Binding – additional information

This section provides additional information for the use of the AQS-TCP binding in "pushed ALI" scenarios.

A.1.1 Session Control Messages

- Sessions are established by the responder entity i.e. the *SessionOpenRequest* message is issued by the responder entity instead of the requester entity. The rest of the session establishment message sequence flows accordingly.

  Note that if *SessionOk* control parameters are to be specified, they should be included in the *SessionOpenRequest* message.

- Session teardown remains as is – the SP will signal to the SC its intention to terminate the AQS session by sending a "going out of service" *aqs:Advisory*. The requester entity then engages the session teardown sequence as defined in the AQS.TCP binding specification. The only difference is that the requester entity need not wait for ongoing requests to complete.

A.1.2 AQS.TCP Tunneling Messages

- *Request* messages are not used
- *Response* messages are used to push XML ALI to the requester entity – as specified in the AQS.TCP binding specification (Section 5).