

NATO STANDARD

AEDP-5

NATO STANDARD ISR LIBRARY INTERFACE (NSILI) IMPLEMENTATION GUIDE

Edition 2

MAY 2013



NORTH ATLANTIC TREATY ORGANIZATION

Allied Engineering Documentation Publication

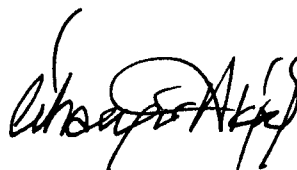
**Published by the
NATO STANDARDIZATION AGENCY (NSA)
© NATO/OTAN**

INTENTIONALLY BLANK

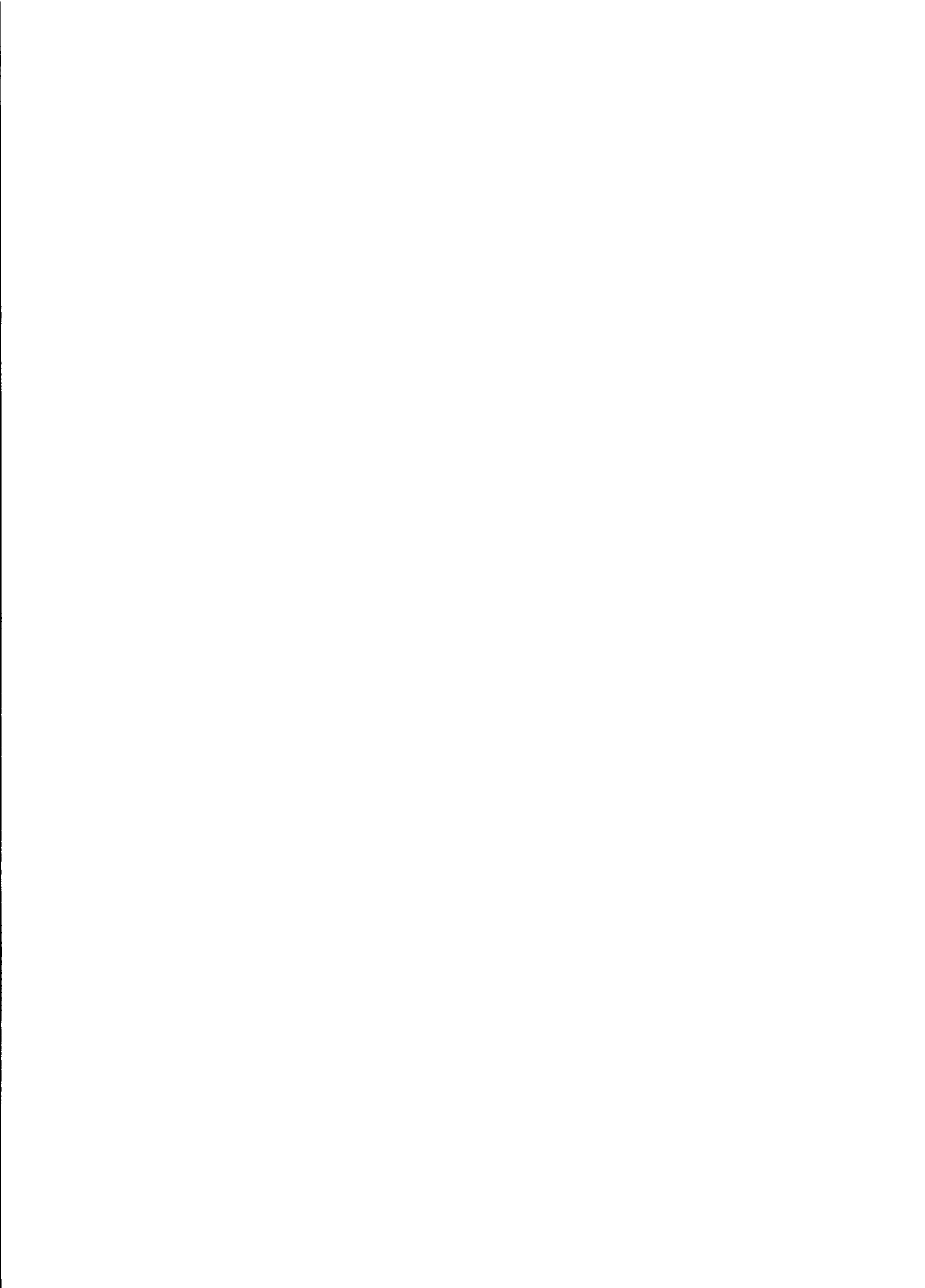
NORTH ATLANTIC TREATY ORGANIZATION (NATO)
NATO STANDARDIZATION AGENCY (NSA)
NATO LETTER OF PROMULGATION

6 May 2013

1. The enclosed Allied Engineering Documentation Publication AEDP-5, NATO STANDARD ISR LIBRARY INTERFACE (NSILI) IMPLEMENTATION GUIDE, which has been approved by the nations in the JCGISR, is promulgated herewith.
2. AEDP-5 (Edition 2) is effective upon receipt/will come into effect on [NED] and supersedes AEDP-5 (Edition 1) which shall be destroyed in accordance with the local procedure for the destruction of documents.
3. No part of this publication may be reproduced, stored in a retrieval system, used commercially, adapted, or transmitted in any form or by any means, electronic, mechanical, photo-copying, recording or otherwise, without the prior permission of the publisher. With the exception of commercial sales, this does not apply to member nations and Partnership for Peace countries, or NATO commands and bodies.
4. This publication shall be handled in accordance with C-M(2002)60.



Dr. Cihangir Aksit, TUR Civ
Director NATO Standardization Agency



Copyright:

(C) Copyright NATO. This document is a derived work from a submission from *NATO Joint Capability Group on Intelligence, Surveillance and Reconnaissance (JCGISR)*. Some rights reserved
- (CC) Attribution-

You are free:

- to copy, distribute, display, and perform/execute the work
- to make derivative works
- to make commercial use of the work

Under the following conditions:

- (By:) attribution. You must give the original author (*NATO NAFAG Air Group IV now the JCGISR*) credit.
- For any reuse or distribution, you must make clear to others the license terms of this work.

Any of these conditions can be waived if you get permission from the copyright holder NATO. Your fair use and other rights are in no way affected by the above. This is a human-readable summary of the Legal Code (the full license is available from Creative Commons <<http://creativecommons.org/licenses/by/2.0/>>).

RECORD OF AMENDMENTS

No.	Reference/date of amendment	Date Entered	Signature

Table of Contents

Glossary of Terms and Definitions	v
Terms and Definitions	v
Acronyms and Abbreviations	v
FOREWORD	ix
1. Introduction	1
2. Aim	1
3. Background	1
4. Goal	2
5. Scope	3
6. Reference Documents	3
6.1 Technical References	3
6.2 Policy and Planning Documents	4
6.3 North Atlantic Treaty Organization Standardization Agreements (STANAGs) and Allied Engineering Documentation Publications (AEDPs)	4
6.4 International Standards	5
6.5 Federal Information Processing Publications	5
6.6 NGA Specifications and Publications	5
ANNEX A: Implementation Guidance	1
A.1. The Development Environment	1
A.1.1 Resources	1
A.1.2 Use of GIAS and UCOS	1
A.1.3 Object Request Brokers	1
A.2. Login Process – Develop LibraryMgr	4
A.2.1 Connecting to a Database	4
A.2.2 NSILI Client/Server Connection	5
A.2.3 Server Set-up and Generation of IOR-Files	5
A.2.4 HTTP(S) servers and their configuration (to be addressed)	7
A.2.5 Security issues for the Server (to be addressed)	7
A.2.6 Download of IORs to NSILI clients	7
A.2.7. Connecting to the NSILI server	8
A.2.7.1 Connecting the Client	8
A.2.7.2 Connections over the Internet	8
A.2.7.3 Login Process	9
A.2.8. Query Process – Develop CatalogMgr	9
A.2.8.1 CatalogMgr - Client Side	10
A.2.8.2 CatalogMgr - Server Side	17
A.2.8.3 Parsing the BQS-Query (refer to Annex F of STANAG 4559, Edition 3)	20
A.2.8.4 Converting BQS-Query to Database –Query	21
A.2.8.5 RequestManager	21
A.2.9 Order Process – Develop Product/ OrderMgr	22
A.2.9.1 Product retrieval	23
A.2.9.2 OrderMgr interface	23
A.2.9.3 OrderRequest interface	29
A.2.9.4 ProductMgr	30
A.2.9.5 AccessManager	31
A.2.9.6 Order Attributes get_parameter_request	31
A.2.10 Requests and Callbacks	31
A.2.10.1 Requests	32
A.2.10.2 Callback	32

ANNEX B: NATO ISR Dissemination Architecture	1
B.1. Overview	1
B.2. Security	1
B.2.1. Community of Interest Released Information	1
B.2.2. Security adaptor layer	3
B.3. STANAG 4559 and Web Services	4
B.3.1. Augmenting STANAG 4559 CORBA Based Libraries with Web Services ...	4
B.4. Multiple Libraries and Bandwidth Management	5
ANNEX C: Test and Certification	1
C.1. NSILI Server Test Suite	1
C.2. Test Venues:	2
C.3. Test Preconditions:	2
C.4. STANAG Compliance:	2
ANNEX D: Configuration Management	1
D.1. Purpose	1
D.2. Scope	1
D.3. STANAG Management Organization	2
D.4. Change Management	4
D.5. Meeting Procedures	9
ANNEX E: Data Models and Metadata	1
ANNEX F: Employment Guidance	1
F.1. Video and Streaming Data	1
F.2. Removing Data from a CSD	1
F.3. United States Imagery and Geospatial Information Systems (USIGS) Architecture Common Object Specification (UCOS) Interface Definition Language (IDL)	2
F.4. CSD Replication Mechanism	3

Table of Figures

Figure 1 – COI’s and Libraries Using the STANAG 4559 Interface	2
Figure 2 – Information Exchange Gateways	3
Figure 3 – Augmenting STANAG 4559 CORBA Based Libraries with Web Services.....	5
Figure 4 – Change Management Process	7
Figure 5 – Change Proposal Form	8
Figure 6 – Consolidated Change Proposal Report and Log Format Example	9

List of Tables

Table A - 1 – Specifying a Chip	25
---------------------------------------	----

Glossary of Terms and Definitions

Terms and Definitions

The following terms and definitions are fundamental to the scope and interoperable implementation of the STANAG 4559 and this AEDP-5. The STANAG 4559 has been developed using several paradigms for access and interaction. The terminology developed has been utilized to describe functionally the role the interface has with the rest of the NIIA and other applications. Terms have also been aligned with the ISO TC211 Terms database which is freely available from ISO Technical Committee 211 at:

<http://www.isotc211.org>

Term	Definition
Client	software component that can invoke an operation from a server (ISO Standard 19118)
Library	The centralized access point for a client to gain functional parts of a system. In password authenticating systems such as the US Image Product Library, this centralized access point allows or denies functional parts. In the product library, there is only one Library Object shared among all users.
Manager	The functional parts of the product library. Managers can be implemented for activities such as query submission and product ordering. In a STANAG 4559 implementation it is possible to share the Managers between all users of the system because no username/password protection is specified.
Requests	The mechanism for completing, tracking, and canceling activities requested of the library. When any activity is performed by a client, the server will issue a request to the client. Three different models of waiting can be utilized by the client to determine if a server activity has completed. These models allow for a wide range of flexibility in how the client is developed. The models are: polling completion, interrupt completion, and blocked completion.
Server	a particular instance of a service (ISO 19128)
Service	distinct part of the functionality that is provided by an entity through interfaces (ISO/IEC TR 14252 (Adapted from))

Acronyms and Abbreviations

The following acronyms are used for the purpose of this agreement.

Acronym or Abbreviation	Definition
AAP	Allied Administrative Publication
AEDP	NATO Allied Engineering Documentation Publication
AL	Attachment Levels
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASIISG	All Source Intelligence Integration Sub Group, the ISRIWG previously served integration of only imagery STANAGs

Acronym or Abbreviation	Definition
BICES/LOCE	<u>B</u> attlefield <u>I</u> nformation <u>C</u> ollection and <u>E</u> xploitation <u>S</u> ystems / Linked Ops-Intel Center Europe
Bi-SC	Bi-Strategic Commands for (NATO) Automated Information Systems. The Bi-SC AIS Core Capability Geographic Services project is aimed at providing NATO's Strategic and Subordinate Commands with Geographic Services, implemented through the Bi-SC AIS Core Services.
BNF	Backus-Naur Form
BQS	Boolean Query syntax
C ⁴ I	Command Control, Communications, Computers and Intelligence
CENTAUR	Cross-Domain Enterprise All-Source User Repository
CAESAR	Coalition Aerial Surveillance and Reconnaissance
CCS	Common coordinate system as used in STANAG 4545
COI	Community of Interest
CORBA	Common Object Request Broker Architecture
CSD	(MAJIC) Coalition Shared Data server / Database
CST	Custodial Support Team
D&R IDM	Discovery and Retrieval Interface Design Model, based on NSG
DAG	Directed Acyclic Graph
DB	Data Base
DCT	Discrete Cosine Transform
DES	Data Extension Segment
DGIWG	Digital Geospatial Information Working Group
DL	Display Levels
FTP	File transfer protocol
GEOINT	Geospatial Intelligence
GIAS	Geospatial and Imagery Access Services specification
GMTI	Ground Moving Target Indicator
HTTPS	Hyper Text Transfer Protocol Secure
IDD	Interface Design Document
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronics Engineers, Inc
IEC	International Electro-technical Commission
IEG	Information Exchange Gateway
IIOP	Internet Inter-ORB Protocol
ILOC	Image Location
IOR	Initial Object Reference
IP	Internet Protocol
IPIWG	(NATO) Intelligence Project Implementation Working Group

Acronym or Abbreviation	Definition
ISAF	International Security Assistance Force
ISO	International Organization for Standardization
ISR	Intelligence, Surveillance and Reconnaissance
ISRPL	ISR Product Library
JCGISR	Joint Capability Group on ISR
JDBC	JAVA Data Base Connectivity
JDK	JAVA Development Kit
JISR	Joint ISR
JNI	JAVA Native Interface
JPIP	JPEG 2000 Interaction Protocol
JWID	Joint Warrior Interoperability demonstration
LAN	Local area network
MAJIIC	Multi-sensor Airborne/ground Joint ISR Interoperability Coalition
Metadata	Structured data about data used to aid the identification, description, location or use of information resources. (Source: Government On-line Metadata Standard)
NAFAG	NATO Air Force Armaments Group
NAT	Network Address Translator
NCIA	NATO Communication and Information Agency
NC3TA	
NGA	(USA) National Geospatial-Intelligence Agency
NIIA	NATO ISR Interoperability Architecture
NMRR	NATO Metadata Registry and Repository
NSA	NATO Standards Agency
NSG	(USA) National System for Geospatial-intelligence
NSILI	NATO Standard ISR Library Interface
ODBC	Object Data Base Connectivity
OMG	Object Management Group
ORB	Object Request Broker
POA	Portable Object Adaptor
POC	Point of Contact
RFC	Request for Change
RFI	Request for information
RRDS	Reduced Resolution Data Set
RTP	Real-time Transfer Protocol
SLOC	Symbol Location
SOAP	Simple Object Access Protocol

Acronym or Abbreviation	Definition
SQL	Syntax query language
STANAG	NATO Standardization Agreement
STGP	Shared Tactical Ground Picture
TRE	Tagged Record Extension
UCOS	USIGS Common Object Specification
USIGS	United States Imagery and Geospatial Information Systems
URL	Uniform Record Locator
VM	Virtual machine
WAN	Wide area network
XML	Extensible Markup Language

FOREWORD

This document updates the original AEDP-5 dated April 2008, and provides the North Atlantic Treaty Organization (NATO) Intelligence, Surveillance and Reconnaissance (ISR) community with technical guidance on developing and testing implementations of the STANAG 4559 NATO Standard ISR Library Interface (NSILI). NSILI is the standard for formatting and exchanging digital ISR data and related products between the libraries of NATO nations. The NSILI Standard is part of a family of Standards that are assembled under NATO Joint Capabilities Group ISR (JCGISR) to ensure the exchange of multi-national intelligence and reconnaissance information.

The NATO JCGISR NSILI Custodial Support Team (CST) developed this document in accordance with current NATO procedures and guidelines under the direction and oversight of the NATO Secondary Imagery Format (NSIF) Custodian. Forward all comments, recommendations, additions, deletions, and other pertinent data that may be of use in improving this document to:

NSILI Custodian
Attn: Laura Moore
National Geospatial-Intelligence Agency
National Center for Geospatial Intelligence Standards
Mail Stop L-66
3838 VOGEL RD
US – ARNOLD MO 63010-6238
Ph: (314) 676 0290
Fax: (314) 676 3015

1. Introduction

Allied Engineering Document Publication (AEDP)-5, Edition 2 provides updated technical and management guidance for implementing the NATO STANAG 4559: NSILI as part of the NATO ISR Interoperability Architecture (NIIA) for ISR systems. STANAG 4559: NSILI is the standard for metadata attribute descriptions and protocol definitions for the discovery and interchange of ISR data among NATO member ISR product libraries.

The aim of the NIIA is to promote interoperability for the exchange of ISR information, including primary and secondary (post-exploitation) imagery, moving target indicator, digital motion imagery and electronic intelligence and signals reports among NATO Command Control Communications, Computers and Intelligence (C⁴I) Systems.

Four levels of interoperability are defined in NATO interoperability publications. The NIIA provides a foundation and means to collect, store and exchange data and adheres to accomplishing the levels of interoperability. The four levels of interoperability are:

- 1 Degree 1: Unstructured Data Exchange. Involves the exchange of human-interpretable unstructured data such as the free text found in operational estimates, analysis and papers.
- 2 Degree 2: Structured Data Exchange. Involves the exchange of human-interpretable structured data intended for manual and/or automated handling, but requires manual compilation, receipt and/or message dispatch.
- 3 Degree 3: Seamless Sharing of Data. Involves the automated sharing of data amongst systems based on a common exchange model.
- 4 Degree 4: Seamless Sharing of Information. An extension of degree 3 to the universal interpretation of information through data processing based on co-operating applications.

It should be noted that the objective of the NIIA is to achieve interoperability at Degree 2. With the implementation of STANAG 4559 and definition of data models /XML discovery and retrieval schemas for the data formats defined in the NIIA, higher degrees of interoperability will be obtained.

2. Aim

This update to AEDP-5, Edition 1 provides technical information that was recognized or developed during the design and interoperability testing of STANAG 4559 implementations during the period 2008-2011. The information found herein was identified as important in designing NSILI implementations that are interoperable in an ISR mission and information exchange network. The AEDP provides guidance on options that may be used in an NSILI implementation and considerations for future developments in the NSILI STANAG.

3. Background

The NSILI Standard (STANAG 4559) specifies a common software interface to be implemented and exist for all NATO interoperable library systems containing ISR data. The interface provides

electronic search and retrieval capabilities for distributed users to find products from distributed libraries in support of, but not limited to, rapid mission planning and operation, strategic analysis, and intelligence preparation of the battlefield. Product Libraries and the NSIL Interface are viewed as a key standards-based technology utilized within existing Request for Information (RFI) procedures.

Within the NIIA, formats for ISR data are defined that include NSIF (STANAG 4545) for multiple still images, text and graphics segments, and the relative orientation of each with respect to the other segments of the image; NSIF also includes provisions for additional types and volumes of data that were not anticipated at the time the format standard was created. Because of this continued expansion of applications of NSIF, the NSILI has been seen to provide a discovery and retrieval capability for a variety of ISR data types. The flexibility of NSILI has been proven in NATO JCGISR and other forums with ISR Ground Moving Target Indicator (STANAG 4607 – GMTI) and Digital Motion Imagery (STANAG 4609), which serve to prove the broader utility of NSILI. The NSILI CST is developing data models based on the JCGISR Metadata Harmonization effort that address interoperable discovery and retrieval methods. Expectations of the NSILI application to future ISR data types serve to further promote this concept. Found within Annex E of STANAG 4559, Edition 3, is a description of the Data Model attribute types and domains; this model will be the reference point for other discovery and retrieval data models as they are defined.

Substantive changes from Edition 2 of STANAG 4559 and Edition 1 of AEDP-5:

- FTP has been disallowed with implementation of Edition 3 of STANAG 4559. FTP data transfer is addressed in Edition 2 of STANAG 4559 and Edition 1 of AEDP-5.
- Edition 3 of STANAG 4559 carries a revised data model developed in concert with the MAJIIC1 project². Later editions of STANAG 4559 data models will be separated from the STANAG and provided as a separate reference/ located on the password protected NMRR website at Hypertext Transfer Protocol Secure (HTTP(S))://nmrr.nc3a.nato.int. Note that at this writing the NMRR website is not fully operational.
- Edition 3 of STANAG 4559 has additionally incorporated the use of HTTPS in accordance with direction from the NATO information security guidance.

4. Goal

The overall goal of NSILI is for users of ISR data to have timely access to distributed ISR information as constraints of operational libraries and security policies permit. The NSILI AEDP

¹ Multi-Sensor Aerospace-Ground Joint ISR Interoperability Coalition (MAJIIC) is a nine nation project, which aimed to maximise the military utility of surveillance and reconnaissance resources through the development and evaluation of operational and technical means for interoperability of a wide range of ISR assets. The nine nations participating in the MAJIIC project are Canada, France, Germany, Italy, Netherlands, Norway, Spain, United Kingdom and the United States of America. The nations have appointed the NATO Consultation, Command and Control Agency (NC3A) as a facilitator for the project and to provide overall technical management. In 2010 NC3A merged with other NATO agencies and the new Agency was renamed the “NATO Communications and Information Agency (NCIA)”.

² The MAJIIC developments regarding STANAG 4559 were an enhancement of the Edition 2 data model and an implementation of the STANAG. These developments led to the Coalition Shared Data (CSD) server, which was part of the deployment within JISR Step1 to ISAF.

supports users and developers by discussing development experiences of existing implementations and providing recommendations for implementation design. Adoption of the NSILI Standard, this AEDP and applicable test tools within Coalition environments, is encouraged by the NSILI CST and the JCGISR as a means to establish and stabilize the interoperability foundation and enable standards-based web-enabled ISR data exchange.

5. Scope

This document includes technical guidance information for developing and testing implementations of NSILI. The sections of this document are as follows:

Annex A: Implementation Guidance

Annex B: NATO ISR Dissemination Architecture Issues

Annex C: NSILI Test and Certification Criteria

Annex D: NSILI Configuration Management Plan

Annex E: Data Models Attribute Types and Domains

Annex F: United States Imagery and Geospatial Information Systems Architecture (USIGS)
Common Object Specification Interface Definition Languages (UCOS IDLs)

Annex G: Employment Guidance

The latest versions of NATO STANAG 4559, the NSILI AEDP-5, Edition 1 and supporting documents are publicly available at: <http://www.nato.int/structur/AC/224/home.htm>. Select “topics” in the left column and then select STANAG 4559 summary. All versions of the Standard and the AEDP-5 can then be selected and downloaded along with supporting documents, ratification status and contact information.

6. Reference Documents

6.1 Technical References

1. MTR 99W: MITRE TECHNICAL REPORT
2. IEEE: Technology of Object-Oriented Languages and Systems September 22 - 25, 1998 Beijing, China
3. Implementation of a Geospatial Imagery Digital Library Using Java and CORBA p. 280
P.D. Coddington, K.A. Hawick, K.E. Kerry, J.A. Mathew, A.J. Silis D.L. Webb, P.J. Whitbread, C.G. Irving, M.W. Grigg, R. Jana, K. Tang
4. OMG Organisation link <http://www.omg.org/technology/corba/corbadownloads.htm>
5. JacORB <http://www.inf.fu-berlin.de/~brose/jacorb>
6. MICO <http://www.mico.org/>
7. ORBacus http://www.iona.com/products/orbacus_home.htm
8. SUN-JDK-ORB <http://java.sun.com/j2se/1.4.2/docs/api/org/omg/CORBA/doc-files/compliance.html>),
9. Java, idl2j, <http://java.sun.com/j2se/1.4.2/>
10. VBOrb <http://home.t-online.de/home/Martin.Both/vborb.html>

11. VisiBroker <http://www.borland.com/visibroker/>

6.2 Policy and Planning Documents

NATO AAP-3 (J) Edition 1, Rev 1, 27 April 2011	Procedures for the Development, Preparation, Production, and the Updating of NATO Standardization Agreements (STANAGs) and Allied Publications (APs)
NATO AEDP-2 Vol 1	Introduction and explanation of the NATO ISR Interoperability Architecture, Sep 2005
NATO AEDP-2 Vol 2	NIIA Management, test and certification guidance, Sep 2005
NATO AEDP-2 Vol 3	NIIA Technical guidance, Sep 2005
NATO AEDP-2 Vol 4	NIIA Terms and Definitions: Sep 2005

6.3 North Atlantic Treaty Organization Standardization Agreements (STANAGs) and Allied Engineering Documentation Publications (AEDPs)

STANAG 4545	NATO Secondary Imagery Format (NSIF) Edition 1, Amendment 1, 14 April 2000
STANAG 4559	NATO Standard ISR Library Interface (NSILI) Edition 3, November 2010
STANAG 4607	NATO Ground Moving Target Indicator Format (GMTI) Edition 3, September 2010
STANAG 4609	NATO Digital Motion Imagery Format (MI), Edition 3, October 2009
STANAG 4633	NATO Signals Intelligence and Electronic Signals Management, working draft January 2005
STANAG 4658 (Study)	NATO Cooperative ELINT/ESM Geolocation (NCEEG) Standard (no date available)
STANAG 4676	NATO ISR Tracking Standard (no date available)
STANAG 7023	NATO Primary Imagery Format, Edition 3, 16 September 2004
AEDP-4	Subject: NATO Secondary Imagery Format (NSIF) Implementation Guide, Edition 1, 2007
AEDP-7	Subject: NATO Ground Moving Target Indicator (GMTI) Format Implementation Guide , Edition 1, April 2008
AEDP-8	Subject: NATO Motion Imagery (MI) STANAG 4609 Edition 3 Implementation Guide, December 2009

AEDP-12 (draft) NATO ISR Tracking Standard (NITS) Implementation Guide (Date TBD)

NSILI Server Test Suite Specification Some validation, test suite and tools are available on the NCIA Collaboration Development Test Evaluation network (NCIA coordination required). Plans call for eventually making the tools available on the password protected MAJIIC portal under the Tools link:
<https://majiic.nc3a.nato.int/MAJIIC/Tools/Forms/by%20STANAG.aspx>

Request access to the test tools through the STANAG 4559 Custodian.

6.4 International Standards

IEEE Technology of Object-Oriented Languages and Systems, Sept 22-25, 1998
ISO 3166 Country Codes

6.5 Federal Information Processing Publications

FIPS 10-4 (sun set status by U.S.) Countries, Dependencies, Areas of Special Sovereignty, and Their Principal Administrative Divisions, April 1995 (Copies of the above FIPS are available on the web at <https://disronline.csd.disa.mil/a/>. Password/CAC required for access.)

6.6 NGA Specifications and Publications

D&R IDM Discovery & Retrieval Interface Data Model, Ver. M, dated 9/30/2007

GIAS Geospatial and Imagery Access Service Specification
National Imagery and Mapping Agency
Version 3.5.1, 6 August 2001

UCOS USIGS Common Object Specification,
National Imagery and Mapping Agency,
Version 1.5.1a, 5 October 2001
http://www.nato.int/structur/AC/224/ag4/4559/4559_UCOS151Approvedplus%20EIT_nu.pdf

MTR 99W

Mitre Technical Report, Geospatial and Imagery Access Services
Specification David P. Lutz, November
1999<http://www.omg.org/docs/gis/99-11-03.doc>

STDI_0002

The Compendium of Controlled Extensions (CE) for the National
Imagery Transmission Format (NITF),
Version 4.0, August 2011

ANNEX A: Implementation Guidance

A.1. The Development Environment

A.1.1 Resources

STANAG 4559, NSILI, is specified using Interface Definition Language (IDL) files of GIAS and UCOS (see next section). These IDL files written using the Object Management Group (OMG) IDL Specification contain the definitions of interfaces, data types and error conditions in a programming language-independent notation. By using appropriate IDL parsers, these files can be readily compiled into CORBA software components for various programming languages including (not exclusive):

- C
- C++,
- Java
- Ada95,
- Smalltalk
- Visual Basic

For the interoperability tests conducted amongst the NSILI CST, the most common language used was Java. There is currently only one client implementation based on C++. As far as it is known, no other languages have been used for an NSILI implementation. This document does not restrict the developer's choice of language.

Several operating systems on different platforms have been tested, to include:

- Windows
- UNIX (Sun Solaris)
- Linux (Redhat and S.u.S.E)
- MAC OS (Macintosh)

This document does not restrict the use of any operating system or platform configuration.

A.1.2 Use of GIAS and UCOS

For NSILI, a subset of the GIAS and the United States Imagery and Geospatial Information System (USIGS) Common Object Specification (UCOS) of the National Geospatial Intelligence Agency (NGA) is used. These documents and the IDL files needed for implementation are available in PDF and HTML from the open NATO Standards web site, along with the 4559 STANAG and this document (see Annex F for more on GIAS and UCOS and a link to the associated documentation).

For further information see references:

A.1.3 Object Request Brokers

For the middleware implementation, various Object Request Brokers (ORBs) are available. See the reference section on OMG's web site for examples of freeware and shareware (link in reference section). Of the variety of Object Request Brokers (ORBs) that are available and have

been used within an NSILI implementation -- CORBA version, GIOP version, IIOP version, and vendor specific versions -- interoperability between different ORBs has been proven during interoperability tests performed between NSILI implementations at various times in the maturation process of the NSIL Interface. See below in this section for the findings of these interoperability tests.

Some of the ORBs and their properties are listed below; those that have been tested for NSILI are highlighted in bold letters:

1. **JacORB** see reference section for links, **Java**, IDL/Java mapping,
2. MICO open source, see reference section for links, C++, IDL2C++
3. ORBacus commercial, fully CORBA 2.4, C++, Java, see reference section for links.
4. **SUN-JDK-ORB** (open source, CORBA 2.3.2 also IDL to Java conversion software **Java**, idl2j see reference section for links.
5. TAO (open source, CORBA 2.2, C++)
6. VBOrb open source, partially CORBA 2.4, Visual Basic, IDL2VB, see reference section for links.
7. **VisiBroker** commercial, CORBA 2.6, C++, **Java**, see reference section for links.

The ORBs highlighted in the list above (JAC-ORB, SUN jdk , VisiBroker) have been tested against each other, and in the process it has been shown that versions of certain ORBs were not interoperable with others or caused problems.. Testing identified the following problems:

General:

An Errata Sheet was posted for Edition 1 of NSILI to modify the use of CORBA 2.4 and use of future CORBA versions was approved provided they were compatible with 4559. The latest version of CORBA is 3.1 (ref: Object Management Group (OMG) Web Site).

VisiBroker:

1. Visibroker's smart agent (osagent), that is an additional, dynamic, distributed directory service, could not be used for newer S.u.S.E Linux operating system versions. For versions higher than 8.2, obviously runtime libraries are missing. No further investigations were made, because the NSILI server could not be started without the osagent.
2. VisiBroker ORB can use both "PERSISTENT" or "TRANSIENT" CORBA-Policies
3. A Virtual Machine (VM) parameter for a server using VisiBroker ORB has to be set to achieve interoperability with ORBs of other vendors:

```
Dvbroker.orb.tcIndirection=false
```

By default this VM parameter is set to "true". In this case a marshalling error is caused when receiving the query results.

Other tests with ORBs, such as the JDK 1.4-ORB, showed some features that could cause marshalling errors.

JDK 1.4.1 ORB

1. When the CORBA Policy is set to “PERSISTENT“ in version *1.4.1* of the Java ORB the CORBA-Policy caused errors. This behaviour could not be explained.
2. The IDL-Parser for the JDK-ORB generates the argument UCO.DAGList for the method complete_DAG_results defined in the interface SubmitQueryRequest, although the argument defined in the IDL file is GIAS.QueryResults. Both types have the same structure. Obviously, the IDL parser is ignoring the preceding type definition. This could be an issue when switching from one implementation using a certain ORB to an implementation using the JDK ORB.
3. To generate the structure DAG_Results, different attribute values have to be inserted into **any** (The purpose of any is to encapsulate data of an arbitrary type) It was observed that attributes (File Length) of the type, UCOS_FILE_SIZE that is a type definition of double, could not be extracted from the corresponding **any** value. Thus for internal use in the NSILI CST test community, attributes of type UCOS_FILE_SIZE were generally set to double for this inserting and extracting process. It appears there is a bug in the JDK CORBA implementation that disallows the use of type definition UCOS_FILE_SIZE. It should be kept in mind that for more sophisticated data models other attribute types might cause equivalent problems.

The ORB validation tests do not intend to restrict the developer to certain versions or platforms but are included for completeness. Examples of ORB instances are displayed below for information only.

Operating System	Windows	UNIX	Windows	Windows	Linux	Windows+Linux
Programming Languages	Java	Java	Java.	C++	Java	Java
ORB	JDK	JDK	JAC-ORB	VisiBroker	JDK	VisiBroker+JDK
Implementations	Client+Server	Client+Server	Client + Server.	Client	Client + Server	Client+Server

A.2. Login Process – Develop LibraryMgr

As described in STANAG 4559, Annex C, there are two primary activities involved in initiating a session of interaction with an ISR library:

- Creation of a database connection to an ISR library or a corresponding digital index (metadata database). This functionality is not part of the NSILI server.
- Set up of an NSILI client/server connection to have access to an ISR library.

Both activities shall be described in the next two subsections.

A.2.1 Connecting to a Database

Although the first activity is not part of NSILI, a short overview is provided to enhance understanding of how ISR databases and other data sources can be accessed. In order to view or to get the data, an appropriate API between the NSILI server and the original database is needed. This API can be designed by using various languages such as Visual Basic, C++ or Java. To avoid the creation of additional interfaces the API should be part of the NSILI server and should use the same language. In fact there are two well-known database connectivity APIs recommended for an NSILI implementation as well as other “object-relational mapping technology tools, such as “Hibernate”:

- Object-relational tools (example; Hibernate)
- Microsoft’s Object Database Connectivity (ODBC) and
- Java Database Connectivity (JDBC).

Hibernate is used for data based connectivity by NATO implementations. ODBC API offers connectivity to almost all databases on almost all platforms and is a widely used programming interface for accessing relational databases. It is applicable for C, C++ and Visual Basic implementations. ODBC cannot be used directly with Java programs. In this case an additional JAVA Native Interface (JNI) is needed. Alternatively the JDBC API can be used. Because many current NSILI implementations are using Java, the basic steps required to handle JDBC are described below as an example:

Step 1	Loading appropriate driver: <code>Use Class.forName("[Path].Driver").newInstance();</code>
Step 2	Establishing a connection: Use the following syntax to get connected to the database: <code>Connection conn = DriverManager.getConnection (String URL,String user, String password)URL= jdbc:<subprotocol>://[hostname][:port]/<subname></code>
Step 3	Creating JDBC statements (these statements are objects that can be executed): <code>Statement stmt = conn.createStatement();</code>
Step 4	Executing the statement (As a parameter an (SQL) query has to be passed. This SQL query is obtained by parsing the corresponding BQS statement): <code>ResultSet rs = stmt.executeQuery("<SQL query>");</code>
Step	Retrieving result sets (several methods are available to

5	loop through the result sets): rs.next(), rs.getInt(), rs.getString() etc.
Step 6	Closing the connection and the statement objects, the close() method is used: conn.close(), stmt.close()

Several interesting features can be derived from this implementation scheme:

- The JDBC API allows a connection to remote databases in a LAN or even a WAN. By selecting port numbers and individual IP addresses, firewall configurations can be adapted.
- Security issues can be taken into account by using user IDs and passwords

A.2.2 NSILI Client/Server Connection

As stated already in STANAG 4559, the networks and communications interfaces should be consistent with the NATO C3 Technical Architecture (NC3TA). NSILI itself describes a client/server application based on CORBA. While Edition 2 of STANAG 4559 (available on NATO Public Web Site) has foreseen FTP for accessing the Initial Object Reference (IOR) string (see below) and accessing directly the products in the library, Edition 3 of STANAG 4559 has changed FTP to HTTP and additionally incorporated the use of secure http (HTTP(S)) in accordance with direction from the NATO information security guidance.

CORBA based programs, implemented in different programming languages, can interoperate on a network of distributed computers using different operating systems. For the communication between client and server, the standard protocol IIOP is used. The following step-by-step description shows what happens each time a client connects to a server system.

STANAG 4559 requires a server to publish the CORBA reference to the Library object as an Initial Object Reference (IOR) string, and make that accessible via http(s). The first necessary step is to exchange a so-called CORBA IOR, which itself was generated by an NSILI server application. This IOR with information about the hardware (IP-address, port number) and the top-level interface (in the case of NSILI, the Library Interface) is converted to a string, which is written to an ASCII file. On the NSILI server side, *the system needs to establish a link to an HTTP(S)*.

For future implementations, the generation of IORs, their download to NSILI client via HTTP(S), appropriate protocols and the selection of parameters shall be described in the next subsections.

A.2.3 Server Set-up and Generation of IOR-Files

To set up the NSILI server and to generate an IOR-File for the client/server binding, several basic initialization steps have to be accomplished. For a better understanding, code snippets for a Java and a C++ implementation are added to the process steps below:

Step 1	<p>Initialization of an Object Request Broker with appropriate arguments:</p> <pre> Java: org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null); C++: CORBA::ORB_var orb = CORBA::ORB_init (argc, argv); </pre>
Step 2	<p>Creation of a "root" Portable Object Adapter (POA) as default POA. The "root" POA is managed by the ORB and is always available to an application through the ORB initialization interface <code>resolve_initial_references</code>:</p> <pre> Java: org.omg.CORBA.Object robj = orb.resolve_initial_references("RootPOA"); POA rootPOA = org.omg.PortableServer.POAHelper.narrow(robj); C++: CORBA::Object_var robj =orb- >resolve_initial_references ("RootPOA"); PortableServer::POA_var rootPOA=PortableServer::POA::_narrow (robj); </pre>
Step 3	<p>Creation of a "child" POA for the top level Interface "Library" with appropriate policies by invoking the <code>create_POA</code> factory operation on a parent POA (here root POA):</p> <pre> Java: POA poaLibrary = rootPOA.create_POA("Library_poa", rootPOA.the_POAManager(), new org.omg.CORBA.Policy[] { ...}); C++: PortableServer::POAManager_var poa_manager = rootPOA->the_POAManager (); CORBA::PolicyList polices (...);policies.length (...); policies0] =...;... PortableServer::POA_var poaLibrary = rootPOA- >create_POA("Library_poa", poa_manager.in (),policies); </pre>
Step 4	<p>Creation and explicit activation of the "Library" servant:</p> <pre> Java: LibraryImpl myLibraryImpl = new LibraryImpl("Library"); poaLibrary.activate_object_with_id("Library".getBytes(), myLibraryImpl) C++: LibraryImpl myLibraryImpl; PortableServer::ObjectId_var Libid = PortableServer::string_to_ObjectId ("Library"); </pre>

	<pre>poaLibrary->activate_object_with_id (Libid.in (),&myLibraryImpl);</pre>
Step 5	<p>Creation of an object reference from a servant:</p> <p>Java:</p> <pre>Org.omg.CORBA.Object objref = poaLibrary.servant_to_reference(myLibraryImpl);</pre> <p>C++:</p> <pre>CORBA::Object_var objref = poaLibrary- >servant_to_reference (&myLibraryImpl);</pre>
Step 6	<p>Generation of an IOR-file:</p> <p>Java:</p> <pre>FileWriter fw =new FileWriter("[path]/<ior-file- name>"); fw.write(orb.object_to_string(objref)); fw.close();</pre> <p>C++:</p> <pre>CORBA::String_var ior =orb->object_to_string (objref); ofstream of ("[path]/<ior-file-name>"); of << ior;of.close ();</pre>
Step 7	<p>Activating the NSILI server:</p> <p>Java:</p> <pre>rootPOA.the_POAManager().activate(); orb.run();</pre> <p>C++:</p> <pre>poa_manager->activate(); orb->run();</pre>

The steps shown above are an implementation example. The definition of policies, the activation of servants (explicit/implicit), or the ways individual POAs are nested with each other are not a subject covered in this AEDP.

To allow client/server binding in accordance with STANAG 4559, only an Initial Object Reference (IOR)-file derived from the Library server has to be provided. This file has to be accessible and downloadable via HTTP(S). Details for this procedure are described in the next subsection.

A.2.4 HTTP(S) servers and their configuration (to be addressed)

A.2.5 Security issues for the Server (to be addressed)

Example configuration for a read and/or write access on a HTTP(S) Server (to be addressed)

These URLs must be made available to the NSILI clients. Depending on the security level, this can be done by the corresponding means of communication (email, phone, fax).

A.2.6 Download of IORs to NSILI clients

In the last two subsections the preparations to establish the client/server connection have been described from the server side. Now we shall have a more profound and detailed view on the client side. The first steps on the client side are to:

- process the server generated URL
- access the corresponding HTTP(S) server and
- download the IOR file.

A.2.7. Connecting to the NSILI server

In this last section the basis for the client/server connection is established.

□

A.2.7.1 Connecting the Client

Three basic steps have to be performed in the client application for a successful server connection. Examples of the steps are given below. On completion of these steps the user should receive a logged on message from the server which will identify the engaged library, and an established connection.

1. The ORB has to be initialized (with additional arguments)

```
Java:
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);
C++:
CORBA::ORB_var orb = CORBA::ORB_init (argc, argv);
```

```
Java:
org.omg.CORBA.Object objRef=orb.string_to_object(ior_string);
C++:
CORBA::Object_var objRef = orb->string_to_object (ior_string);
```

2. The IOR string has to be converted to a CORBA object

3. The CORBA object is narrowed

```
Java:
Library mylibrary =GIAS.LibraryHelper.narrow(objRef);
C++:
Library::Library_var mylibrary =Library::_narrow(objRef.in());
```

A.2.7.2 Connections over the Internet

To initiate a CORBA connection to a TCP firewall protected server in an Internet environment, the client has to use defined port numbers and the firewall IP-address. This firewall IP address has to be mapped to an internally known, private address of the server on the server side, using a Network Address Translator (NAT). NATs allow a single device, such as a firewall or a router, to act as an agent between the Internet and a local (or "private") network. This means for a client that only a single, unique IP address is required to access even several different servers in a LAN. The router (firewall) then maps this IP-address to the predefined server. The specification for NATs can be found in the Request for Change (RFC) 1631.

There are two essential problems when trying to use the Internet Inter-ORB Protocol (IIOP)

across today's firewalls:

- Location-transparency and the dynamic allocation of addresses as done by CORBA middleware make it difficult to know in advance the host and port addresses used for transactions.
- Addressing information contained in an object reference is invalidated when crossing a Network Address Translating router.

The first problem can be avoided by using fixed IP addresses and communication ports. As already described for HTTP(S), all this information has to be known in advance to get the right firewall configuration. Using appropriate IOR files that contain both the "official" IP address and the internal "private" IP address can solve the second problem. Further information and guidance is available through the OMG CORBA Firewall Transversal Specification
<http://www.info.fundp.ac.be/~ven/CIS/OMG/firewall%20traversal.pdf>

Other Comments

The Java ORB has special settings to address some of the issues.

For the orbd program the option: "-port NNN" fixes the port used for communicating with the nameserver to a given port.

For the orbd program the option:

"-JDcom.sun.CORBA.ORBServerHost= remotely_known_host" publishes the given remote host (or IP address, as known outside the firewall), instead of an internal IP address of the server.

A Java CORBA program can be started with the option:

"-Dcom.sun.CORBA.ORBServerHost=remotely_known_host" to publish the remote host in the IOR string and other CORBA references.

A Java CORBA program can be started with the option:

"-Dcom.sun.CORBA.POA.ORBPersistentServerPort=NNN" to fix the port used for communication between client a server to a given port.

A.2.7.3 Login Process

Many interoperability tests and experiments have been performed between the nations' implementations within the last several years. A successful login process has proven to be the key for interoperability. Usually minor changes for the other interfaces were necessary when appropriate configurations and codes were used to establish the FTP³/HTTP(S) and CORBA connections. This is true for connections in Local Area Network (LAN) and Wide Area Networks (connections over the Internet).

A.2.8. Query Process – Develop CatalogMgr

This part of the AEDP describes implementation details for the CatalogMgr. For a detailed description of the functionality refer to [STANAG4559] and [GIAS351]. The description provided in this Annex is divided into a Client side and a Server side description. The description templates are written in Java but should be applicable easily to any implementation language.

³ FTP is no longer supported in STANAG 4559 Edition3.

Pseudo Code is written as Java Comment.

A.2.8.1 CatalogMgr - Client Side

The CatalogMgr holds the functionality of querying the Catalog of the ISR-Library. For this purpose, it has 2 methods:

- hit_count returns the amount of hits for the specified query via HitCountRequest. (2.8.1.3) Counting hits with hit_count()
- submit_query returns the result of the specified query via SubmitQueryRequest. (2.8.1.5) Searching Catalog with submit_query()

Both methods use the GIAS.Query Struct, see [GIAS351]

It should be understood, that a subsequent call of hit_count() and submit_query() could theoretically have a different amount of results, if the database is modified between calls.

A.2.8.1.1 CatalogMgr Reference

As basis for further operation the Client has to get a CatalogMgr Reference. During the Login-process the Client requested and received a Library-Reference, which will enable the request of all managers associated with the Library. The code sample below shows how a CatalogMgr Reference request is achieved in Java.

```
Client.java: getting CatalogMgr Reference

LibraryManager mgr =
Library.getManager("CatalogMgr");
CatalogMgr catalogManager =
CatalogMgrHelper.narrow(mgr);
```

A.2.8.1.2 Counting hits with hit_count()

The Client can determine the number of results ("hits") that would be returned from the Query. A successful invocation of this operation returns a reference to a HitCountRequest object. If invocation fails the following exceptions are raised:

Exception	Meaning
UnknownViewName	ViewName in aQuery is unknown.
BadQuery	aQuery is syntactically invalid
BadQueryAttribute	aQuery contains an attribute unknown to the CatalogMgr

Further Experiences on these exceptions:

Unknown ViewName: The ViewName is mandatory. It can be asked from the CatalogMgr for

instance as `CatalogMgr.get_view_names()` or must at least be hard coded as “PRODUCT_DEFAULT_VIEW”..

Bad Query: This is mostly an error generated by the server’s BQS-Parser. The BQS-Query is incorrect.

Bad QueryAttribute: Mostly: ViewName and Attributename do not match. See Section 3.2.5 (The `GIAS::Query` struct).

A.2.8.1.3 HitcountRequest

The *HitCountRequest* is returned by a successful invocation of the *hit_count* operation of the *CatalogMgr*. It provides the operation `complete()` to retrieve the count of the results of the submitted query. The possible States of *HitCountRequest*, which the Client can see, and their meaning is displayed below. Further operations concern the calls of the *RequestManager* (see Section (2.8.5)).

State	Description
PENDING	Initial state
IN_PROGRESS	Query started
COMPLETED	result is available
ABORTED	Request aborted Result is not valid
CANCELED	Request canceled Result is not valid

A.2.8.1.4 HitcountRequest.complete()

This operation allows a client to complete processing of the *HitCountRequest*. This operation blocks until the requested operation reaches an end state. A successful invocation of this operation (State “COMPLETED”) returns a value that indicates the total number of results (“hits”) that would be returned if the query was executed.

```

Client.java: hit_count() (synchronous call)
org.omg.CORBA.IntHolder hitCount = new
org.omg.CORBA.IntHolder();
GIAS.HitCountRequest hitRequest=null;
UCO.State hitState=null;
GIAS.Query aQuery= ...;
try {
    hitRequest = CatalogManager.hit_count(aQuery, new
NameValue[0]);
    ...
}
try {
    hitState = hitRequest.complete(hitCount);
    ...
}
return hitCount.value;

```

Using Callbacks is optional and can be denied by the server (register_callback will fail). Further descriptions for the use of callbacks and what has to be taken into account can be found in Section 2.8.2.5 Requests and Callbacks of this Annex. Callbacks can be useful on long-time operations and asynchronous events.

Client.java: Creating a callback (tbd)

```

Client.java: hit_count() (using callback)

    org.omg.CORBA.IntHolder hitCount = new
org.omg.CORBA.IntHolder();
GIAS.HitCountRequest hitRequest=null;
UCO.State hitState=null;
GIAS.Query aQuery= ...;

    try {
        hitRequest = CatalogManager.hit_count(aQuery, new
NameValue[0]);
        ...
    }
    Callback cb = getCallbackObject (hitRequest);
    String CallBackId =
hitRequest.register_callback(cb);

    if (CallBackId != null)                // Server
accepts Callbacks

    {
        waitForNotify(); // do something else ...
        hitRequest.freeCallback(CallBackId);
    }
    Try
    {
        hitState = hitRequest.complete(hitCount);
        ...
    }

```


A.2.8.1.5 Searching Catalog with submit_query()

This operation allows a client to submit a query to the CatalogMgr.
This operation needs as input parameters:

- The GIAS.Query.
- The resultlist tells the server which Metadata to provide
- The SortAttributeList for sorting the results
- The Propertylist defaults to “WGS84”

The type of resultlist is from type UCO.NameList. The Client tells the server, which Metadata the client wants to have. The list is a subset of the Metadata of the selected View in the Query. The Client can get the complete list with DataModelMgr.get_attributes(). If the list is empty, then no metadata attributes will be returned just the UID.Product.

The SortAttributeList tells the server, how and by which attributes the results have to be sorted. This list may only consist of sortable attributes. If the list is empty, then the order is server implementation specific.

A successful invocation of this operation will return a reference to a *SubmitQueryRequest* object described below.

If invocation fails, standard exceptions are raised, as described in Table 8

A.2.8.1.6 SubmitQueryRequest

The *SubmitQueryRequest* is returned by a successful invocation of the *submit_query* operation of the *CatalogMgr*. It provides operations to retrieve the results of the submitted query in two forms: as a DAG (Direct Acyclic Graph) or as a XML-Document. This interface defines the following operations:

- set_number_of_hits()
- complete_DAG_results()
- complete_XML_results()

If complete_StringDAG_results() is not supported and the server will raise a *CORBA.NO_IMPLEMENT*-Exception. . Further operations concern the calls of the RequestManager (see Section (2.8.5)). Possible states of SubmitQueryRequest and their meanings are displayed below:

State	Description
PENDING	Initial state
IN_PROGRESS	Searching started
RESULTS_AVAILABLE	First Results available, results are not completed
COMPLETED	Server has all results completed
ABORTED	Request aborted Results are not valid
CANCELED	Request canceled Results are not valid

```

Client.java:  submit_query()

    String[] results = null;
    GIAS.SubmitQueryRequest subRequest=null;
    UCO.State subState=null;
    GIAS.Query aQuery= ...;

    try {
        subRequest = CatalogManager.submit_query(aQuery,
            results,sort,props);
        ...
    }

```

A.2.8.1.7 Set_number_of_hits

The operation `set_number_of_hits` allows a client to delimit the number of results (“hits”) that are returned by the invocation of one of the “complete“-operations.

The number of hits should be geared to the required format for the results.
The value entered is valid just for the actual `SubmitQueryRequest`.

```

Client.java: (client side) set_number_of_hits

    Unsigned long hits = ...;
    GIAS.SubmitQueryRequest subRequest=...;
    try {
        subRequest.set_number_of_hits (hits);
        ...
    }

```

A.2.8.1.8 complete_DAG_results

The Client completes processing of the `SubmitQueryRequest` with this operation... A successful invocation returns a `UCO.DAGList` structure containing results from the query. The number of results contained in this list depends from the history of the `submitQueryRequest` and the server implementation.

- If the client has set number of Hits, the number will be less or equal this limit.
- Otherwise all results or a server-internal limit of results will be returned.

The call has to be repeated, until the return list is empty.

If the operation is cancelled or aborted, the results in the list are no longer valid.

```

Client.java: complete_DAG_results() (synchronous)

    State Status=null;
    GIAS.DAGListHolder dresults = new DAGListHolder();
    GIAS.SubmitQueryRequest subRequest=...;

    try {
        Status = subRequest.complete_DAG_results(dresults);
    if ((number_of_results < number_of_hits) || (number_of_results
    == 0)) // all results retrieved
    else // further results available
    }

```

A.2.8.1.9 Complete_XML_results

The Client completes processing of the *SubmitQueryRequest* with this operation.. A successful invocation returns a *UCO.XMLDocument* structure containing results from the query. The number of results contained in this list depends on the history of the *submitQueryRequest* and the server implementation.

- If the client has a set number of Hits, the number will be less or equal this limit.
- Otherwise all results or a server-internal limit of results will be returned.

The call has to be repeated, until the return list is empty.

- If the operation is cancelled or aborted, the results in the list are no longer valid.
- As there is no XML Schema defined at the moment, the resulting XML-Document is Library Specific.

```

Client.java: complete_XML_results (synchronous)

State Status=null;
GIAS.StringHolder xresults=new StringHolder();
GIAS.SubmitQueryRequest subRequest=...;
    try {
        Status = subRequest.complete_XML_results(dresults);
        if ((number_of_results <
number_of_hits)|| (number_of_results == 0))
            // all results retrieved
        else // further results available
    }

```

A.2.8.1.10 Building the BQS-Query

The following examples are provided as guidance to the developer in constructing a BQS query string that expands on the discussion of the BQS query examples given in STANAG 4559, Annex

F.

A.2.8.1.11 Some Examples of the BQS-Query with NSIL_ALL_VIEW

Query for date and time:

"NSIL_FILE.dateTimeDeclared > '2005/10/09 12:00:0.0'" looks for all products younger than specified date

Same query, attribute given as fully qualified attribute name:

"NSIL_PRODUCT:NSIL_FILE.dateTimeDeclared > '2005/10/09 12:00:0.0'"

Using wildcards:

"(NSIL_IMAGERY.comments LIKE '%bridge%' or NSIL_IMAGERY.comments LIKE '%Munich%')" looks for products with an NSIL_IMAGERY comments, which contains the words "bridge" or "Munich". The round bracket closes the logical "or", could be omitted in this case.

Using text comparison:

"NSIL_IMAGERY.category = 'SAR' or NSIL_IMAGERY.category = 'VIS'" looks for products whose NSIL_IMAGERY category matches exactly 'VIS' or 'SAR'.

Geospatial query (1)

"spatialGeographicReferenceBox inside RECTANGLE (60.0 , 5.0 , 5.0 , 60.0)" looks for products, whose spatialGeographicReferenceBox is completely inside the specified rectangle (upperleft , lowerright). Coordinates are in degrees (latitude, longitude).

Geospatial query (2)

"spatialGeographicReferenceBox intersect POLYGON (55.0 , 5.0 , 55.0 , 9.0 , 52.0 , 9.0 , 52.0 , 5.0 , 55.0 , 5.0)" looks for products which are completely or partially inside specified polygon. Arrangement of the coordinates is counter clockwise. Coordinates are in degrees (latitude, longitude).

Geospatial query (3)

"spatialGeographicReferenceBox within 20000 meters of POINT (48.0 , 11.5)"; looks for products, whose spatialGeographicReferenceBox is completely inside a circle of 20 km round specified point. Coordinates are in degrees (latitude, longitude).

A.2.8.1.12 Converting Boolean Query Syntax (BQS) to Structured Query Language (SQL) Syntax

Though it may be possible to convert BQS to SQL syntax with only string replacements, implementers have found it easier to use a parser generator to create a parser for BQS, and use the parse-tree to build the SQL.

[STANAG4559] contains the Backus Naur Form (BNF) notation for the BQS in Annex F. Writing the input for a parser generator based on this BNF notation is relatively straightforward.

Note: There can be problems related to parsing the BQS using a generated parser. For example, valid operators and values are dependent on the type of attribute. A straightforward parser will not be able to validate the query completely for correctness.

Solution: After creating the parse tree with the parser, the parse tree can be checked for incompatible attribute-operator and attribute-value pairs.

Mapping geospatial operators to SQL is highly database specific. There are Implementations that use Oracle DB with Spatial module (Oracle 11g has geospatial module embedded). Other implementations use PostgreSQL DB with PostGIS and Microsoft SQL Server 2008 with geospatial support.

A.2.8.2 CatalogMgr - Server Side

The CatalogMgr implements the functionality of querying the Catalog of the ISR-Library on Server Side. For this purpose, it has 2 methods:

- hit_count returns the amount of hits for the specified query via HitCountRequest.
- submit_query returns the result of the specified query via SubmitQueryRequest.

There are several common topics:

- Parsing the BQS-Query
- Convert BQS-Query to Database –Query
- Connecting to RequestManager

When creating the CatalogMgr server implementation all users are able to share the same CatalogMgr instance because in the STANAG 4559 text the choice was made to ignore the access criteria. If a server chooses to also allow access criteria along with “anonymous“ usage that is required by STANAG 4559, the server may need a CatalogMgr instance per user in order to hold the access criteria.

The code provided in the following examples is intended to illustrate some of the logic that has been applied to the methods. [Per NGA44] [M.D45]

A.2.8.2.1 hit_count() - Server side

On server side the hit-count()-invocation has to create a HitCountRequest, add it to the RequestManager and return it to the caller. HitCountRequestImpl is the Implementation of the interface.

```
Server.java: CatalogMgr.hit_count()

GIAS.Query aQuery= ...;
    try {
        HitCountRequestImpl sv = new
HitCountRequestImpl(catalogMgr, aQuery);
        org.omg.CORBA.Object ref = orb.activate_object(sv);
        HitCountRequest request =
HitCountRequestHelper.narrow(ref);
        requestManager.addRequest(sv);
        ...
    }
    return request; // return HitCountRequest
```

A.2.8.2.1.1 HitCountRequest

The HitCountRequest analyzes the Query with the BQSParser converts it to the Database Query and executes the Query. If analysis or execution fails, appropriate exceptions have to be raised.

```
Server.java: HitCountRequest.

        BQSParser bqp = new BQSParser (aQuery);
        attributeDefs = bqp.analyze(); // analyzes
BQS-Query
        DatabaseQuery dq = new DatabaseQuery (attributeDefs);
        // Create Database-Query
        sq.execute(); // Executes Query
        State = State.COMPLETED;
        if (use_callback)
            send_notify();
```

A.2.8.2.1.2 State.COMPLETED

The server has to check, whether the execution of the Query has finished. This means the HitCountRequest has reached an “end state” (COMPLETED, ABORTED, CANCELED). The server will wait for an “end state” to be reached. If the Query was successful (State.COMPLETED), the result is read, Result and State are returned. Alternatively if it is allowed to postpone the DBQuery until complete() is called, considering that it can be quite fast. The state would immediately be set to State.COMPLETED

If the Query reaches State.ABORTED or State.CANCELED.....

```
Server.java: HitcountRequest.complete()

    while (!State.isAnEndState())
        Wait();
    If (State == State.COMPLETED)
        number_of_hits.value = getNrOfHits ();    // get result
    return state;                                // return state
```

A.2.8.2.2 Submit_query() - Server side

On server side the submit_query()-invocation has to create a SubmitQueryRequest, add it to the RequestManager and return it to the caller. SubmitQueryRequestImpl is the Implementation of the interface.

```
Server.java: CatalogMgr.submit_query()

    GIAS.Query aQuery= ...;
    try {
        SubmitQueryRequestImpl sv = new
SubmitQueryRequestImpl(...);
        org.omg.CORBA.Object ref = orb.activate_object(sv);
        SubmitQueryRequest request =
SubmitQueryRequestHelper.narrow(ref);
        requestManager.addRequest(sv);
        ...
        return request;                // return
SubmitQueryRequest
```

A.2.8.2.3 SubmitqueryRequest

The SubmitQueryRequest analyzes the Query with the BQSParser converts it to the Database Query and executes the Query. If analysis or execution fails, appropriate exceptions have to be raised.

The execution of the query by the SubmitQueryRequest can be done at several opportunities. If the query is expected to take a short amount of time it may be executed even before the CatalogMgr returns the SubmitQueryRequest to the client. The state will be COMPLETED immediately when the client receives the request reference.

The incoming results of the query are stored in a unique structure, such as an UCO.NameValueList or a HashMap. If using callbacks, SubmitQueryRequest has to notify the client when the number of results exceeds the set_number_of_hits() or as execution is completed.

A.2.8.2.4 Complete_XXX_results() (to be completed)

The server has to wait, until one of the following conditions is fulfilled:

- SubmitQueryRequest has reached an end State (COMPLETED, ABORTED, CANCELED).
- SubmitQueryRequest is in State RESULTS_AVAILABLE and the number_of_results is greater or equal number_of_hits.

If State is COMPLETED, the number of returned results is counted as follows:

- If number_of_hits is set, number_of_hits or the rest of the results will be returned
- If number_of_hits is not set, but the server has an internal limit, the limit or the rest of the results will be returned.
- Elsewhere all remaining results will be returned.

If State is RESULTS_AVAILABLE, number_of_hits or the rest of the results will be returned.

```

Server.java: SubmitQueryRequest.complete_XXX_result()

while (true)
{
    if (State.isAnEndState())
        break;
    if ((State == State.RESULTS_AVAILABLE) && (no_res >=
no_hits))
        break;
    wait();
}
If (State.isAnEndState() && State == State.COMPLETED)
{
    If (limit.isSet())
        If (limit < no_res)
            Results[] = getResults (limit);
        else
            Results[] = getResults (no_res);
    else
        Results[] = getResults (no_res);
}
Else
    Results[] = getResults (limit);
return state;

```

A.2.8.2.5 Building the DAGList-Result

The DAG lists are defined in the GIAS and USIGS Common Objects Specification (UCOS) Supporting Documents found at: <http://www.nato.int/structur/AC/224/home.htm>. Select topics in the left column and then select Standards, and then AEDPs to find the reference documents.

A.2.8.3 Parsing the BQS-Query (refer to Annex F of STANAG 4559, Edition 3)

A.2.8.4 Converting BQS-Query to Database –Query

A.2.8.4.1 Converting BQS to SQL Syntax

Though it may be possible to convert BQS to SQL syntax with only string replacements, implementers have found it easier to use a parser generator to create a parser for BQS, and use the parse-tree to build the SQL.

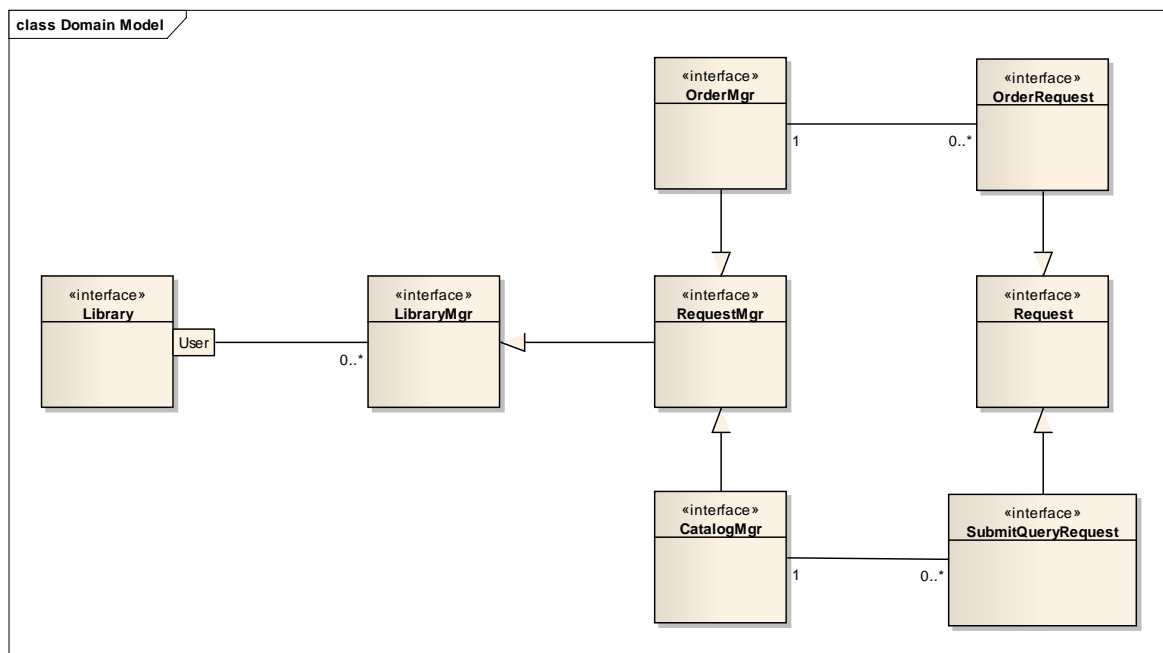
STANAG 4559 contains the BNF notation for the BQS in Annex F. It is relatively straight forward to write the input for a parser generator based on this BNF notation.

Note: There can be problems related to parsing the BQS using a generated parser. For example, valid operators and values are dependent on the type of attribute. A straightforward parser will not be able to validate the query completely for correctness.

Solution: After creating the parse tree with the parser, the parse tree can be checked for incompatible attribute-operator and attribute-value pairs.

Mapping geospatial operators to SQL is highly database specific. There are implementations that use Oracle DB with Spatial module (Oracle 11g has geospatial module embedded). Other implementations use PostgreSQL DB with PostGIS and Microsoft SQL Server 2008:

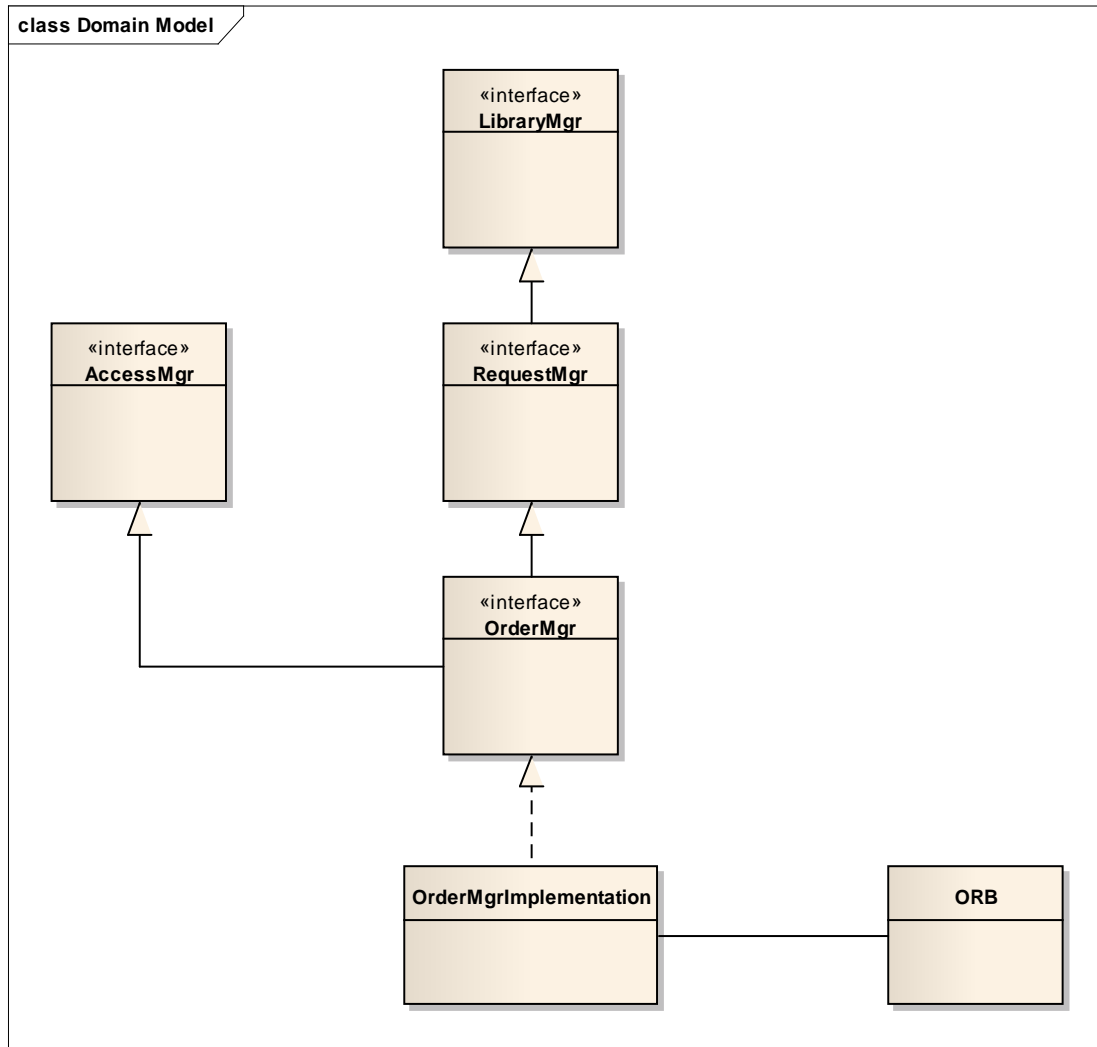
A.2.8.5 RequestManager



Whenever a user logs in into the Library (by calling `get_manager`) a new manager instance of the requested type is created, unless there is already an instance of that manager for the user. Therefore, users will only see their own managers and they are not able to look into the library managers of other users. As defined in GIAS, all managers, other than the `DataModelMgr`, derive

from the RequestMgr interface, which is derived from the LibraryMgr interface. The managers derived from the RequestMgr interface contain associated requests returned by the methods of that manager. As shown in the image above, the OrderMgr has associated OrderRequests, and the CatalogMgr has SubmitQueryRequests (HitCountRequests are not shown for simplicity reasons). A manager can only know about those requests it created (by an invocation of one of its methods). A manager will never know about requests created by another RequestMgr.

As a result, all of the methods of the RequestMgr interface that take a Request as a parameter can only use requests generated by this specific RequestMgr. All requests created by other RequestMgrs are unknown to this RequestMgr and will return an UnknownRequest exception.



As displayed above, the ORB references to the implementation object is available to the outside world through its implemented interfaces. As a result, the OrderMgr implementation can be viewed as a LibraryMgr, as a RequestMgr and as an AccessMgr. On a CORBA level, all three interfaces are accessed using the same Initial Object Request (IOR) address.

A.2.9 Order Process – Develop Product/ OrderMgr

A.2.9.1 Product retrieval

After querying on product metadata there are several possibilities to retrieve the product itself, as well as files related to the product.

- Order product on-line using the OrderMgr, delivery by HTTP(S) or e-mail, Order product with alteration using the OrderMgr (e.g. only part of an image, or chip)
- Order off-line products like paper maps using the OrderMgr.
- Request related files to be delivered by HTTP(S) using the ProductMgr.
- Direct retrieval by a URL given in a metadata attribute of the product or related files.

All libraries must support product delivery via the OrderMgr. Support for alterations, and off-line products will depend on the library implementation.

Libraries supporting related files must support delivery using the ProductMgr.

A.2.9.1.1 Comparison of direct access versus ordering

Retrieving a product using a URL is by far the easiest to implement retrieval method in a client. The NSIL_ALL_VIEW view for example contains the NSIL_FILE.productURL attribute. It will contain a URL. In most modern programming languages retrieving a URL is very easy.

Several limitations exist with using this URL linkage. No conversion, packaging, or chipping can be specified in this notation. Also, multiple products cannot be retrieved to a machine other than the client.

A URL allows many different protocols to be used. In general libraries should use only “HTTP”, “HTTPS” protocols for file-based direct access. Client implementers should note that other protocols could be used. For example a video stream could be identified by an “RTP” protocol. Another example could be a JPIP protocol for JPEG2000 images.

A URL allows specification of a communication port number. This provides flexibility on the server side, e.g. to have two libraries (test and production) running on the same system, but providing products on a different port. Network and firewall must be considered because they must allow routing of network packets from the required port.

Ordering using the OrderMgr and related file retrieval via the ProductMgr will in contrast use the HTTP protocol to send the files. The OrderMgr also allows sending files using e-mail or a hardcopy or physical media was ordered and will be delivered physically. The advantage of the OrderMgr is its many packaging and conversion options. For example, products can be very large images (hundreds of megabytes). The OrderMgr allows selecting only a subset or chip of the image, saving big on bandwidth (but requiring more processing power on the server).

The OrderMgr and ProductMgr send files to an HTTP(S) server (push), while the direct access URL allows a client to pull files. The OrderMgr and ProductMgr require the client to have access to an HTTP(S) server where they can send the files to.

A.2.9.2 OrderMgr interface

The OrderMgr provides many options to allow a flexible ordering process:

- Alterations: chipping, scaling, changing image compression, changing image mode.
- Different delivery protocols: HTTP(S), e-mail, physical

- Different packaging options: separate files or packaged in a ZIP or TAR file, with or without file compression.

Appendix 1 of Annex D of [STANAG4559] is mostly dedicated to the ordering options and possible values to use in ordering fields. The following sections detail implementation issues for the OrderMgr not given in the STANAG.

A.2.9.2.1 Image Chipping

Image chipping is the process of creating a subset or chip of an image. Libraries can contain very large images (hundreds of gigabytes). An operator may be interested only in a particular area. Instead of retrieving the whole image to the client, the OrderMgr can chip the area of interest from the large image and send only that smaller image to the client. Currently the only type of subsection allowed is rectangular (re: GIAS dated 6 August 2001, Pg. 8, Paragraph 2.2.1.2).

There are several advantages: The client may not be able to deal with very large images (due to memory or processing restrictions); in networks with limited bandwidth transferring the full image may not be feasible.

A.2.9.2.1.1 Specifying a chip

The main data structure to specify a chip is GIAS.ImageSpec. Table D-1- 6 in Appendix 1 of Annex D of [STANAG4559] details the possible field values. A chip is specified using the sub_section and geo_region_type fields. Refer to [GIAS351] for details of the GeoRegionType type. For products with multiple image segments, the imageid field allows selection of the single segment to chip (see also section below on multi-image products).

Table A-5 contains a description of the chipping options.

geo_region_type	sub_section	Comments
ALL	Irrelevant	No chipping, use full image (all segments)
LAT_LON	x = longitude, y = latitude in decimal degrees	Chip using geographic coordinates. Will only work for images with geo-location information. It is a rectangle with top and bottom parallel to the image top and bottom.
LINE_SAMPLE_FULL	Coordinates in the coordinate system of the full resolution image (the product).	Don't use NSIF CCS, but single image coordinates. Requires knowledge of full image size
LINE_SAMPLE_CHIP	Coordinates in the coordinate system of the OVERVIEW related file.	Allows selecting the area of interest using the smaller OVERVIEW file. Server needs to know scale factor of OVERVIEW file.
NULL_REGION	Irrelevant	No chipping, no image.

Table A - 1 – Specifying a Chip

In one typical use case, assuming a large image product, a client could first retrieve the related file of type “OVERVIEW”. The user can visually select the area of interest. The client will use the GeoRegionType.LINE_SAMPLE_CHIP value to indicate the selected coordinates are based on the OVERVIEW related file.

In another typical use case the user may have expressed an area of interest and the client automatically orders chips of products overlapping the geographic boundaries of the area of interest. The GeoRegionType.LAT_LON would be used in this case.

A.2.9.2.1.2 Overview related file

The related file of type OVERVIEW must be an NSIF file. Because chipping will only chip a single image segment (see section on multi-image products for more details), it is necessary that the OVERVIEW related file reflects the same structure as the NSIF product file. This includes:

- Same number of image segments
- Image segments in same order
- All image sub-headers included.

In summary it must be a scaled down version of the NSIF product file.

[STANAG4559 calls for a maximum image dimension of 1024x1024 for the OVERVIEW related file to allow the client display to fully present the image without sub-sampling. For easy scaling, the OVERVIEW related file could be a “reduced resolution image” (usually this is an R5 level Reduced Resolution Data Set (RRDS) as specified in [NSIF-AEDP-4] Annex E. This implies a scale factor that is a power of 2, which can be expressed in the IMAG field in the image sub-header. When using another scale factor, the ICHIPB Tagged Record Extension (TRE) must be included to accurately specify the scale factor.

Keeping the same block-masks and padding pixels as the full image segments can be useful in some cases. Chipping is sometimes limited to a single block, so knowing the block structure can be useful if the overview is used to define a chip.

In other cases keeping the overview file small may be a requirement, and using C8 (JPEG 2000) compression is desired, but C3 (JPEG Discrete Cosine Transform (DCT)) may be acceptable. In keeping many small blocks this may cause a lot of processing overhead. In such cases joining all blocks may be desired.

Note that it is not always possible for a client (nor for the server) to calculate the size of the image in the product file from the size of the image in the OVERVIEW related file. Many times it is possible taking into account the IMAG sub-header field and ICHIPB TRE. The product file might be a chip itself, so the IMAG in the product file may not be 1, and it could already have an ICHIPB TRE. So the client cannot always use the GeoRegionType.LINE_SAMPLE_FULL when ordering a chip based on the OVERVIEW file, but should use the GeoRegionType.LINE_SAMPLE_CHIP.

The server must know the scale factor used to produce the OVERVIEW related file when a chip

is ordered with LINE_SAMPLE_CHIP. It could either store the scale factor in a file or database, or recompute it from the image size in both the OVERVIEW and product files.

A.2.9.2.1.3 Chipping multi-image products

NSIF image files can have multiple image segments. The important thing about chipping of multi-segment products is to understand how systems create products of this nature and their use of the Display Levels (DL) and Attachment Levels (AL) along with the use of the Image Location (ILOC) and the Symbol Location (SLOC). For example multiple image segments could be used for:

- Several partly overlapping photographs that together form a larger picture (e.g. of a coast line).
- Annotating images or icons.
- Multiple distinct images, possibly with different geospatial coordinates, having a logical rather than a positional relation.

Chipping multiple image segments (all those that intersect with the chipping area) adds a lot of complexity. There are even some cases where it is impossible to define the meaning of the order contents when dealing with multiple image segments.

Given the complexity and possible problems in correctly interpreting the resulting NSIF file of a multi-segment chip, it is recommended only to chip single image segments.

The ImageSpec structure contains a field "imageid". This is intended to indicate which image segment from a multi-segment NSIF file should be chipped. The imageid is numeric; it is taken

To be considered for imageid definition:

- If imageid is empty, the image segment with attachment level 0 and the lowest display level is used.
- If imageid is numeric, it is taken to be the sequence number of the image segment, starting with 0 for the first image segment. (alternative would be the display level).
- If imageid is not numeric, the first image segment with IID1 field equal to the imageid is chosen.

to be the sequence number of the image segment, starting with 0 for the first image segment within a multi segment file.

A.2.9.2.1.4 Side note on chipping multiple segments

An NSIF file can contain multiple images, but also multiple graphic and text segments containing additional annotations. The given chipping recommendation means ignoring all those. This section contains some more details on the problems to be solved when chipping multiple segments.

Some possibilities with multiple images e.g.:

- Join all image segments (and possibly also graphics segments) into one image and chip that. The resulting image will be of the requested size (including cutting off graphic segments on the chipping borders). The usability of such an image is questionable for strategic analysis but is usable as a tactical picture, because most metadata, including detailed location and orientation, are lost.
- Chip all images individually, assuming the chip coordinates are relative to the Common Coordinate System (CCS) of the NSIF file.
- Exclude image segments that are completely invisible (or include them with an empty image so the metadata (such as comments) is kept. But, in that case the CCS may be larger than the chipping bounds.
- Or include only the main image

For files with Graphic segments e.g.:

- Include all graphic segments
- Include only graphic segments that are completely within the chipping bounds
- Include only graphic segments that are at least partly within the chipping bounds (but this will mean the resulting CCS will be larger than the chipping bounds).
- Try to chip the graphic segments to exclude elements of a graphic segment outside the chipping bounds. But this might cripple the graphics so much they cannot be comprehended anymore.
- Exclude all graphic segments

For Non-graphical segments like Text or DES e.g.:

- Exclude all
- Exclude DES but include Text, e.g. if the purpose of chipping is to limit bandwidth and a DES segment contains a large audio (or worse, video) recording, then including the DES segment defeats the purpose of chipping.
- Include all
- Don't allow chipping if there is a DES segment that is not an overflow.

Below is a list of problems to be encountered when dealing with multi-segment image chips.

- A problem may be that one of the intersecting images was attached to a non-intersecting image. The best thing to do in this case is to not include this intersecting image if the area of interest is not part of the intersection. If it is part of the intersection then it may require user intervention. Remove all segments (also graphic and text) attached to a segment that is removed as a result of chipping. Annotating images and icons could be misinterpreted if they would be chipped (be only partly visible). So only include image segments other than the main one that are completely within the bounds of the chipping area. This strategy will obviously not work well for NSIF files with partly overlapping photos. The question is whether the annotated images are useful without the graphic or text segments. So the graphic and text segments should be considered for inclusion. Some graphics may be related to part of an image segment that is not in the chip.
- In general the chips should be put together as one would for a single image merge from multiple images with possible overlaps in the CCS.
- If chipping with GeoRegionType.LAT_LON, and the NSIF file contains multiple image segments with geospatial location (IGEOL field), each image segment must be chipped separately.

- Image segments (e.g. annotated images) without IGEOLO fields cannot be chipped with GeoRegionType.LAT_LON and must be removed from the chipped file.

A.2.9.2.2. Sub-header fields in a chipped image

When creating the image chip, some file and image sub-header fields need to be changed:

- NROWS, NCOLS: Reflect the size of the chipped image.
- PVTTYPE, IREP, and others: If besides chipping also a different image mode or compression is requested in by the order.
- IGEOLO: when present it must be recomputed to represent the new geospatial bounds of the image. It must be recomputed even when an ICHIPB TRE is added or changed (contrary to e.g. the BLOCKA TRE, which must not be recomputed when the ICHIPB TRE is used).
- CLEVEL: The chip may be much smaller than the original NSIF file. It will contain just one image segment, while the original NSIF file could have had several. These factors may influence the CLEVEL.
- FDT: Represents the time when the file with the chip was created.
- FL, HL, NUMI, etc: The File Length, Header Length, number of segments, and related fields all need to be recomputed.
- ONAME: Name of the Originator which can include library identification.
- OPHONE: . Contact details of file Originator.

Other image sub-header fields must not be changed, specifically:

- IDATIM: defined as the date the image was acquired. See also [NSIF-AEDP-4], Annex A, section 23.
- Since chipping does not change the acquisition time, IDATIM must not be changed.

A.2.9.2.3 Extended and user defined sub-headers in chipped images

The Compendium of Controlled Extensions [STDI-0002] contains useful information including TREs for imagery in [STDI-0002]. Another reference is the [NSIF-AEDP-4], Annex C section 5.5.15 and Annex D; section 4.1.30 & 4.7 on reduced resolution imagery.

Several TREs contain acquisition parameters needed to correctly interpret or geolocate the image. In NSIF images the IMAG field in the image sub-header indicates whether those TREs where recomputed in a reduced resolution image. The ICHIPB TRE is more relevant for chipping. Its presence indicates that other TREs where NOT recomputed. The information in the ICHIPB TRE allows a client to compute the coordinates in the chipped image back to the original image, and thus to the other TREs that contain information relative to the original image.

For NSIF image segments there are two options when creating a chip:

- a. To recalculate and update all metadata in the TREs and not include an ICHIPB tag, or
- b. To leave the existing metadata in the TREs unchanged and add the ICHIPB tag.

It is recommended to always use the ICHIPB TRE, so the other TREs can be included in the chipped image without change. Repeated recalculation of metadata could deteriorate the quality

of the metadata.

If the original image already contained an ICHIPB TRE, the data contained in it must be recomputed to reflect the new chip-of-a-chip. The offsets and magnification must refer to the full image size, because the other TREs refer to it. Note that the original full size unchipped image need not be available in the library. The ICHIPB TRE carries enough information to correctly interpret the other TREs.

Some support data in TREs relate to scan blocks within a single blocked image segment. Chipping can cause some scan blocks to not be included in the chip, invalidating support data. The ICHIPB TRE can only relate back to a single original scan block number. [STDI-0002] therefore recommends not chipping across block boundaries.

A.2.9.2.4 Naming of file(s) in the delivery package

STANAG 4559, Edition 3, Annex D, Appendix 1, section D 3.1.2-1-3 contains a description of the file naming strategy.

Specifying multiple HTTP(S) locations with a non-empty file_name in at least one FileLocation and ordering separate files should not be allowed, because that could result in differently named files at each HTTP(S) site, and no possibility to report this back in the DeliveryManifest.elements structure. The DeliveryManifest.packet_name must be filled in, this field should contain the product filename which could include its extension e.g .tar or .zip.

A.2.9.2.5 Reduced sized images

One alteration the OrderMgr can do is create a reduced sized image. In [STANAG4559] Table D-1-6, the ImageSpec.rrds field is intended for this.

- An rrds of 0 indicates the image is at normal size.
- A value of 1 requests the image at half of the normal width and height.
- A value of 2 requests the image at a quarter of the normal width and height.
- For each further level the width and height need to be divided by 2.

See also [NSIF-AEDP-4], Annex D (NSIF approved Support Data Extension Listing), e.g. on setting the IMAG sub-header field correctly.

Multiple rrds levels could be requested, since the rrds field is a list. The intent is to ship multiple files, one for each requested level. The DeliveryDetails for the product should contain multiple file names, one for each requested level.

A.2.9.3 OrderRequest interface

A.2.9.3.1 OrderSize in request_details

The OrderRequest.request_details method must provide, among others, an OrderSize. The OrderSize may not be accurately known until the order is completed, especially when the order will be a zipped file or must do chipping.

The OrderSize in the request_details need not be static. It could contain a best guess initially, and change over time when a better guess can be made. E.g. start with the total of all file sizes.

Use 0.0 to indicate the OrderSize is not known.

A.2.9.3.2 HTTP(S) Server for Order Manager

Using the OrderMgr the client needs to have access to an **HTTP(S)** server. Clients could set up an **HTTP(S)** server on their own system, allowing direct file access to the delivered package.

Some clients may not be able to have a local **HTTP(S)** server. They can use any other **HTTP(S)** server they have access to. The server would deliver the package there, and the client needs to retrieve it. This solution may double the network traffic, because the package needs to go over the line twice.

The **HTTP(S)** server could be local to the NSILI server. In this case the package needs to be only once on the network. A disadvantage is the additional administrative work on the NSILI server side to maintain **HTTP(S)** accounts for all clients.

A.2.9.4 ProductMgr

The ProductMgr Interface provides operations that allow the client to determine characteristics of a specific product and retrieve related files associated with that product, such as thumbnail and overview images.

A.2.9.4.1 GetParametersRequest

Clients can access product metadata attributes in two ways:

1. Pass the desired result attributes to the CatalogMgr.submit_query method.
2. Request only the product reference via the CatalogMgr and use the ProductMgr.get_parameters method to get the attributes.

Besides requesting a list of attributes, a few special names can be passed to the ProductMgr.get_parameters method:

- ALL: request all metadata attributes available for the product. This will include attributes that are not queryable.
- ORDER: Attributes relating to ordering. This could include both server capabilities with respect to ordering, such as supported transformations, and product metadata needed for ordering, such as the size of the image. Order attributes are further discussed in section 2.9.6 of this document

A.2.9.4.2 GetRelatedFilesRequest

Related files like a thumbnail or overview can be retrieved using the ProductMgr. Libraries may also provide direct access URLs in the metadata as an alternative retrieval method. In the NSIL_ALL_VIEW view the NSIL_RELATED_FILE.URL field is intended for these direct access URLs.

The location parameter of the ProductMgr.get_related_files method provides the hostname, username and password, and directory path where the server can copy the related files. It is assumed, though not explicitly documented in the STANAG, that the **HTTP(S)** protocol with standard **HTTP(S)** ports is used to deliver the files, just as the OrderMgr delivers files by

HTTP(S).

While the list of Related File Types was at one point to include footprint, text, xml, reccexrep, recon4, video, html and audio the fact that there is no specification for these file types caused the CST to not include them in the list of standard related file types in [STANAG4559] Appendix 1 Annex D. The CST does recognize that these and other file types will be found in an ISRPL and should be acceptable to a knowledgeable NSILI implementation.

The ProductMgr can be used to provide alterations, by using a naming convention for the related file type. The capability will be limited to predefined transformations, and usually these transformations have been performed by the server at ingest rather than being done on request. But it should be pointed out that the related file types need to be standardized, or at least well-understood between client and server for the results to be reliable.

For example the related file type “OVERVIEW_JPEG” could provide the overview image in JPEG format instead of NSIF format. “PRODUCT_R1” could indicate a related file in NSIF format at half the original size.

A.2.9.5 AccessManager

The ProductMgr and OrderMgr implement the AccessManager interface. The AccessManager methods are really only useful for orders. The ProductMgr also implements it because it was anticipated that some systems might store thumbnails or overview images (or other related files) off-line. But most systems store those on-line.

NSILI clients should therefore prefer use of the OrderMgr to call the AccessManager methods.

In [STANAG4559] the ProductAccess use_mode has been removed to make it clear that using the AccessManager methods on the ProductMgr is not useful.

Since the server still needs to implement the AccessManager methods in the ProductMgr, it may choose to implement them with the same code as the OrderMgr.

A.2.9.6 Order Attributes get_parameter_request

In order to access metadata of artifacts, the client can either make a call to the search interface (CatalogMgr/StandingQueryMgr) of the library or access the ProductMgr. When accessing the ProductMgr, it is required that the UID:Product reference for the product is available to the client before invoking the ProductMgr. Such a reference can be received from previous search operations or from an ingest of a product.

The parameter “desiredParameters” indicates the attribute which shall be contained in the returned upon calling “complete” on the returned request.

The DAGs returned by this operation are identical to the DAGs returned by the search interfaces (CatalogMgr/StandingQueryMgr). Note that the operation only works with UID:Product references that point to an actual product. UID:Product references pointing to associations will not return the metadata of the association.

A.2.10 Requests and Callbacks

A.2.10.1 Requests

A.2.10.1.1 Deleting a request

The effect of calling `RequestManager.delete_request` for a request that is not in an end-state, such as `PENDING`, `RESULTS_AVAILABLE` or `SUSPENDED` state, should be to first cancel the request, then delete it. Because the `CANCEL` state will not trigger a notify method on the callback, the client, when using a callback, will only see a `Callback.release`.

A.2.10.1.2 Request.cancel versus RequestManager.delete_request

Both `Request.cancel` and `RequestManager.delete_request` will stop request processing and indicate that no more results are needed. `Request.cancel` will keep the `Request`, and thus the resources associated with it, until a timeout period has passed. `RequestManager.delete_request` will allow the server to immediately delete the request.

The client should not expect any results to be available after calling `cancel`. The server is allowed to free resources associated with a request, except those needed during the timeout period, after a call to `cancel`.

The only reason to use `Request.cancel` is to be able to use the request object to get the state. For example when a client uses a request, and the server, or a server administration client wants to cancel a request for some reason, they can use the `cancel` method. The client will be able to check the request state to know the request was cancelled and not normally ended. Normally clients should use `RequestManager.delete_request` to free server resources as soon as possible.

A.2.10.1.3 SUSPENDED state

The `SubmitStandingQueryRequest` and `SubmitQueryOrderRequest` have methods `pause` and `resume` to make the request go in and out of the `SUSPENDED` state. Other requests do not have such methods. Still [GIAS351] Appendix G indicates the `SUSPENDED` state for other requests. Requests other than `SubmitStandingQueryRequest` and `SubmitQueryOrderRequest` are not required to implement the `SUSPENDED` state.

Advanced servers may put a request in a `SUSPENDED` state, e.g. when the request is inactive and the server wants to temporarily swap it to disk to free memory.

A.2.10.2 Callback

A.2.10.2.1 SubmitQueryRequest callback and state interpretation

When using callbacks with a `SubmitQueryRequest`, the client expects the server to start processing the request and notifying the client when finished (or in the case of the `RESULTS_AVAILABLE` state when it has enough results to allow the client to call the `complete_XXX_results` method without blocking). The server should not wait for a call to `SubmitQueryRequest.complete` before starting to process the request.

If the client has not called the `set_number_of_hits` method, the server must choose some number of hits to be able to go in the `RESULTS_AVAILABLE` state. Some implementations will never

hit the RESULTS_AVAILABLE state, because their initial number of hits is very high. Those implementations will immediately go to the COMPLETED state. The state diagram for SubmitQueryRequest in [GIAS351] allows such a state change from start to IN_PROGRESS to COMPLETED without ever hitting RESULTS_AVAILABLE.

A more common situation could be where the number of results is less than the number of hits per invocation of complete_XXX_results. Before getting enough results to trigger RESULTS_AVAILABLE all results have been computed, and the request goes to the COMPLETED state.

A.2.10.2.2 Callbacks are optional

For callbacks, it bears mentioning that both clients and servers that are supporting callbacks should consider failure recovery mechanisms. Both systems need to be able to gracefully handle all conditions that could reasonably occur regarding system availability.

Since callbacks are optional, clients should also be able to fall back to synchronous retrieval of results, in case a server is not supporting callbacks.

A.2.10.2.3 Preventing deadlocks when implementing callbacks

Implementations have found that attention should be given to the synchronous nature of CORBA calls. For example, consider the following pieces of (pseudo) code:

Client1.java: (client side)

```
callback = new Client1CallbackImpl();
org.omg.CORBA.Object ref = activate_object(callback);
synchronized(callback) {
    // Attempt to make sure the callbackId is set in our
    callback
    // before the server calls it. This may cause a deadlock.
    String callbackId =
request.register_callback(CallbackHelper.narrow(ref));
    callback.setCallbackId(callbackId);}
```

Client1CallbackImpl.java: (client side)

```
public synchronized void setCallbackId(String
aCallbackId)
{
    callbackId = aCallbackId;
}
public synchronized void _notify(UCO.State theState,
UCO.RequestDescription description)
{
    log("Called callbackid " + callbackId + " ...");
    ...
}
```

SomeRequestImpl.java: (server side)

```
public String register_callback (Callback acallback)
throws InvalidInputParameter, ProcessingFault, SystemFault
{
    ...
    if (alreadyCompleted) {
        acallback._notify(getState(), getDescription());
    }
    ...
}
```

Here the client tries to make sure the logging in the callback `_notify` method will print a valid `callbackId`. But the server, on registration of the callback immediately calls back the `notify` method (synchronously). Since the client is holding a flag on the callback object, the server call has to wait until the client releases the flag, but the client is waiting for the server to return from the `register_callback`. In other words this is a deadlock situation.

Below is another example, this time deadlocking the server.

Client2.java

```
private SubmitQueryRequest queryRequest = null;

public void run()
{
    queryRequest = catalogMgr.submit_query(...);
    callback = new CallbackImpl();
    org.omg.CORBA.Object ref = activate_object(callback);
    queryRequest.register_callback(CallbackHelper.narrow(ref));
}
...
private class CallbackImpl
{
    public synchronized void _notify(UCO.State theState,
        UCO.RequestDescription description)
    {
        ...
        if (theState == UCO.State.COMPLETED) {
            ...
            queryRequest.complete_DAG_results(...);
        }
    }
    ...
}
```

SubmitQueryRequestImpl.java: (server side)

```
public synchronized String register_callback (Callback acallback)
throws InvalidInputParameter, ProcessingFault, SystemFault
{
    ...
    if (alreadyCompleted) {
        acallback._notify(getState(), getDescription());
    }
    ...
}
// Synchronized to prevent multiple concurrent calls
public synchronized complete_DAG_results(...)
{
    ...
}
```

While registering the callback, the client is called back by the server already, and immediately tries to get the results. But since the server is holding the flag within the `register_callback` method, the `complete_DAG_results` method is waiting for the release of the flag, which will

never happen, i.e. a deadlock.

Solution:

1. Implementers of NSILI clients must be aware of this, and try not to synchronize callback code.
2. Implementers of NSILI servers must either make the callback asynchronous or use a separate thread to do the callback, thus not blocking the `register_callback` method.

In Java the callback can be done asynchronously using code like:

```
org.omg.CORBA.Request request = cb._request("notify");
Any stateAny = request.add_in_arg();
StateHelper.insert(stateAny, getState());
Any requestDescriptionAny = request.add_in_arg();
RequestDescriptionHelper.insert(requestDescriptionAny,
getDescription());
request.send_oneway();
```

The disadvantage of asynchronous calls is that there is no way to get exceptions back. Opening the port seems to still be synchronous, so you will get an exception if the client cannot be reached. For example, with a firewall between a client and server that didn't allow the connection to be made back from the server to the client, the server will hang for some time in the `send_oneway` method until the attempt times out. CORBA debugging showed that it was trying to open the port on the client.

ANNEX B: NATO ISR Dissemination Architecture

B.1. Overview

The NSILI is a key element in the NIIA, which in turn supports the NATO ISR dissemination Architecture. By adhering to the NSILI guidelines, participating Nations create an effective dissemination network that allows them to both populate and access each other's libraries. The end result of this effort is the creation of a robust environment that enables seamless and effective data exchanges that optimize warfighting capability. However, before the NSILI, or the network itself can be fully effective, issues such as security and releasability must be addressed.

B.2. Security

STANAG 4559 assumes that security issues are handled "outside" of the API, as described in the STANAG **Aim** section:

*"The overall goal is for the users, who may be intelligence analysts, imagery analysts, cartographers, mission planners, simulations and operational users from NATO countries, to have timely access to distributed ISR information **if Host Nation operational restrictions and security policies permit this access**".*

This view is reinforced in the ANNEX B; Key Assumption d: *"The NSIL interface assumes that servers and clients are installed on System High networks with network and workstation protections appropriate to the classification level of the network. It is assumed that all information loaded into an NSIL library is authorized for access by all users authorized for the network. Neither NSIL libraries nor clients add additional protections beyond that provided by the network and workstation security protections of the operational network".*

Note: Some implementers use organic applications that test the releasability markings for consistency and if an inconsistency is noted the data is not ingested. Future test cases should be designed to test for this capability.

GIAS, which STANAG 4559 is based on, supports usage of an *AccessCriteria* object which could be passed when getting a manager e.g. the *CatalogMgr*. In STANAG 4559 it was chosen to not use the *AccessCriteria*. To allow security policies within the STANAG 4559 API, one could employ the GIAS *AccessCriteria* mechanism. Introduction of *AccessCriteria* into a future version of the STANAG will affect server and client implementations, but it is assumed that this can be done with relatively few changes.

Employing the *AccessCriteria* mechanism is not the only option for implementing security policies with STANAG 4559. Another option would be to build security mechanisms *around (or on top of) libraries using the STANAG 4559 interface*. Two such mechanisms are discussed in the following subsection:

- 1) Community of Interest (COI) released information
- 2) Security adaptor layer

B.2.1. Community of Interest Released Information

One approach for looking at security is to look at the “space” of information sharing. STANAG 4559 has traditionally focused on one *all-inclusive* “information sharing space”. In reality, information sharing architectures could often be configured as multiple interconnected subsets of the whole space. The subsets are so-called “Communities of Interest” (COI).

A “Community of Interest” (COI) is defined as a dynamic group, set up to perform or support a mission, operation or activity. A COI may also be a stable group such as “HQx / J2 – Intel”, though it may also be formed ad hoc for a specific mission.

A COI is a “space” where information is shared. As such it is homogeneous in terms of security; i.e. all COI participants are in the same security domain. Each COI should have a designated CSD that ensures it has the global metadata set for all products to be shared within the network.

It can be assumed that the different COIs will filter the information they expose/release to other COIs exemplified by the BICES/LOCE network. Libraries using the STANAG 4559 interface could be used as the information exchange mechanism between these COIs, as illustrated in Figure 1 below.

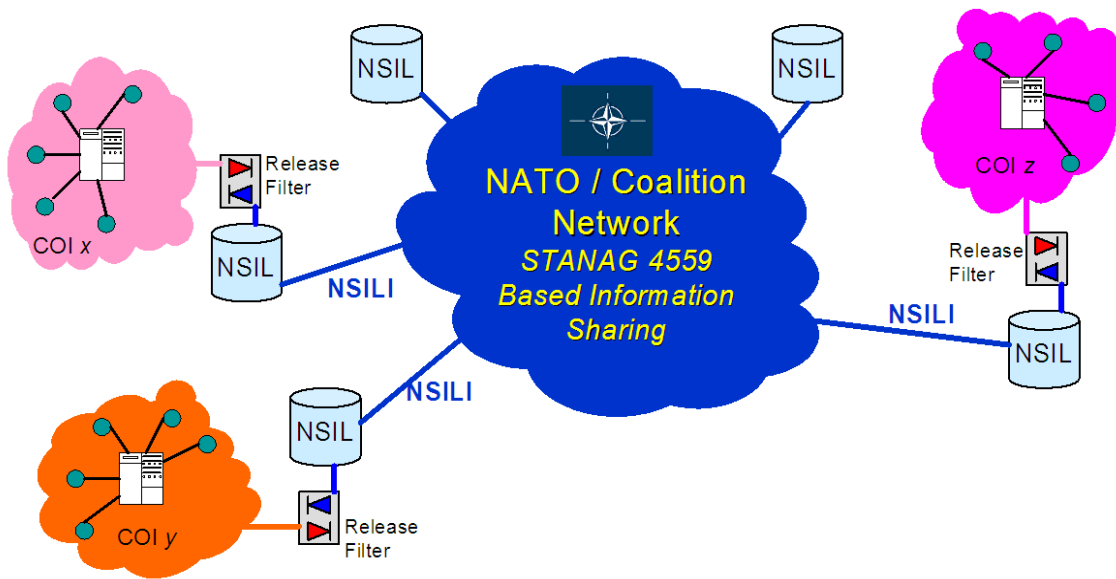


Figure 1 – COI's and Libraries Using the STANAG 4559 Interface

Architectures as shown in the figure above will allow the different COIs to have one or more STANAG 4559 libraries within the COI where all clients will have access to all information. At the same time the COI can dedicate one STANAG 4559 library as the “exchange” library where only information approved for release outside the COI will be posted. When the information has been released to the exchange library, the original STANAG assumption that all information loaded into the library is available for access by all authorized users still applies.

The approach of using STANAG 4559 libraries to expose released information from COIs to the coalition has been demonstrated in the Multi-sensor Aerospace-ground Joint ISR Interoperability Coalition (MAJIIC) project Cross - Domain Enterprise All-Source User Repository (CENTAUR). Similarly, Canada has employed a configuration where a CSD is designated as a Border CSD that exposes released information to the external COI enabling the retrieval of products across the border while disguising the product source location.

B.2.2. Security adaptor layer

Another approach for working with Communities of Interest would be to move the “exchange libraries” on the inside of the release filter. This is illustrated in Figure B-2 Information Exchange Gateways and Libraries Using the STANAG 4559 Interface below.

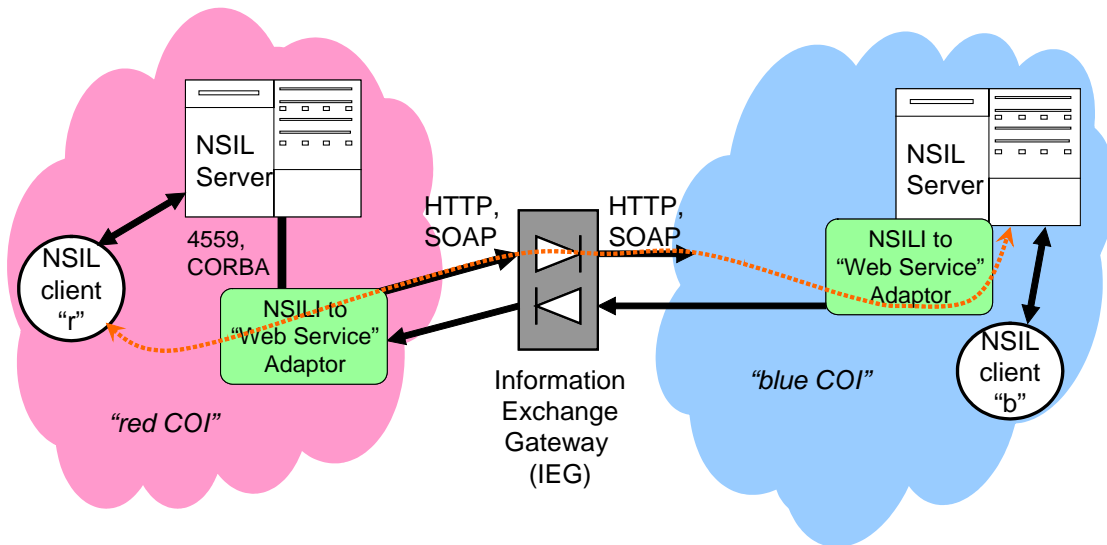


Figure 2 – Information Exchange Gateways

Release filters often includes usage of firewalls, which typically will prevent an NSIL client on the outside of the COI from communicating with an NSIL server on the inside using the CORBA-based STANAG 4559 interface. The main reason for this obstacle is that CORBA requires more ports to be opened than a strict firewall configuration will allow. One way to solve the problem of restrictions in the “release filter” is to use a different communication protocol between the COIs (e.g. pure HTTP, or a Web Service using SOAP, which needs only one port and is ASCII character based).

For clients residing within the same COI as the server, normal STANAG 4559 interfaces between clients and servers will apply. The original STANAG assumption that all information loaded into the library is available for access by all authorized users still applies as long as clients and servers are within the same COI.

Instantiations of this notional architecture can take several forms. Depending on how advanced the adaptor layer is designed, security features like authentication and integrity could be implemented between the COIs. It would also be possible, based on authentication, to grant different clients different levels of product access.

An example of one such architecture was demonstrated by NC3A at the Joint Warrior Interoperability Demonstration (JWID) in 2004. In the JWID scenario, clients on the “blue” COI were querying an NSIL server on the “red” COI by communication over HTTP through an “HTTP-to-NSIL” adaptor residing on the inside of the red COI. Similarly clients on the red COI were

communicating with the blue COI server.

The “release filter” employed at JWID was the so-called Information Exchange Gateway (IEG). The IEG is a NATO developed concept to achieve a means to concentrate (point of presence) and secure information exchange between NATO Bi-Strategic Commands Automated Information System (Bi-SC AIS) Automated Information Systems and Command and Control systems that reside in external domains.

- This security architecture, by tailoring the “adaptors”, should in theory be configurable to support any security requirement.
- security architecture, by tailoring the “adaptors”, should in theory be configurable to support any security requirement. cept to a in external COIs.
- security architecture, by tailoring the “adaptors”, should in theory be configurable to support any security requirement.

B.3. STANAG 4559 and Web Services

The current trend in software development is to employ Web Services as part of system architectures, and a question that needs to be answered is ‘how Web Services relates to STANAG 4559’. There are two radically different ways to look at Web Services and STANAG 4559:

- 1) Web Services can be employed as a complementary technology to the CORBA based STANAG 4559. Special services could be implemented using Web Services technology that could add value to STANAG 4559 based architectures:
 - a. It could add more advanced security capabilities using authentication and integrity checks as described in section B.1.2 above.
 - b. It could potentially allow “new client implementations” simplified access to information in STANAG 4559 libraries (assuming the Web Service adaptors provide very basic functionality), and hence provide a potentially quicker path to interoperability.
- 2) Web Services can be used to build a completely new and alternative architecture from scratch for sharing ISR information, employing different querying mechanisms, ordering mechanisms, etc. This would basically mean developing a new STANAG.

It is not within the scope of the AEDP to enter an analysis of pros and cons of Web Service based architecture versus CORBA based architecture, nor if there are any benefits in choosing to re-implement a library sharing mechanism using Web Services. Because of the number of legacy systems that exist using the STANAG 4559 API, and because the STANAG 4559 API is a mature and proven technology, this document will only look at Web Services as a technology for adding value to STANAG 4559 architectures and not as a replacement technology. Future versions will implement web services as appropriate.

B.3.1. Augmenting STANAG 4559 CORBA Based Libraries with Web Services

Integrating Web Services with a STANAG 4559 client is the approach chosen by the NATO/Coalition project Shared Tactical Ground Picture (STGP). In STGP, the content of a STANAG 4559 based MAJIC Coalition Shared Database (CSD) is exposed through Web Services.

Providing Web Services interfaces to STANAG 4559 CORBA based libraries can be implemented as adaptor layers directly on the STANAG 4559 server, or as adaptor layers on

STANAG 4559 clients, as shown in Figure B-7 Augmenting STANAG 4559 Libraries with Web Services below.

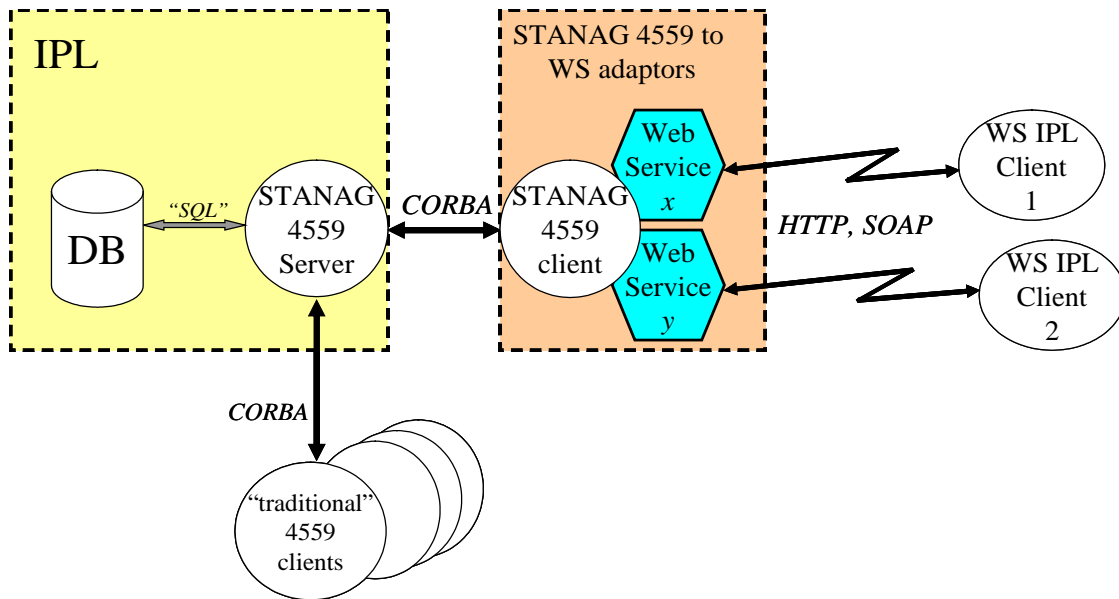


Figure 3 – Augmenting STANAG 4559 CORBA Based Libraries with Web Services

It is probably both simpler and cleaner to integrate the Web Service adaptors with the STANAG 4559 client software, than integrating directly with the server software.

B.4. Multiple Libraries and Bandwidth Management

There will be scenarios on inhomogeneous networks where available bandwidth will vary between network nodes. Figure B-5 COI's and STANAG 4559 Libraries of this Annex could be seen as an example of such a network. In such networks clients in one network node might need to access libraries residing on different network nodes. A consequence of such architectures is that one particular product might travel multiple times between two nodes (e.g. two clients in node A asks for the same image from an IPL in node B). This multiple transmission of the same product between nodes is unnecessarily reducing the overall available bandwidth.

Solutions to solve the problem of duplicate transmissions over a wide area network are outside the scope of STANAG 4559, but this AEDP will provide a short overview of potential approaches.

There are at least three different approaches that could be chosen to address this problem. Each of the three approaches assumes that there is at least one STANAG 4559 library per network node, and each of the solutions assumes that the clients only connect to one server (a local server residing on the same network node).

Alternative solutions:

- 1) Metadata only replication with product caching: All metadata is replicated/synchronized between libraries on all nodes. After the first client on a node pulls down a product, the product is cached within the network node. If a second client on the same node requests the same product it is fetched from the local network cache.

- 2) Full product replication: All products are replicated between all libraries on all nodes. This means that clients will always find every product at their local library. A penalty of this approach is that everything is replicated whether it is needed at a node or not. This may lead to wasteful bandwidth usage.
- 3) Search broker with partial product caching: The search broker will transparently perform queries on all libraries known to the broker on behalf of its client. After a client has downloaded a product, it might be cached as long as the client uses direct access to the product and is not using the OrderMgr.

Assuming that the typical scenario will have library contents that consist of products that are mostly of a size that is significantly larger than the metadata needed to describe these products, and; where clients frequently request products from servers residing in remote nodes, the preferred solution is probably the first one - "metadata only with product caching". Product caching will have a better effect on saving bandwidth compared to the third solution with partial product caching since it will also allow caching of products ordered through the OrderMgr. If the assumption that frequent requests to servers in remote nodes is not correct, the third solution might be an equally good solution.

The second solution is not a viable alternative if there are any large-sized products in the libraries, which there typically will be given that images and video clips are normally large-sized files.

One consequence of employing the metadata replication solution is that the server software must be extended such that the OrderMgr can function as a proxy to the remote library OrderMgr in cases when the product resides with a remote library. In addition to this extension, a mechanism for ingest of metadata from remote libraries must be implemented, and each product must be universally and uniquely identifiable in the metadata. In other words, metadata must only be received from the "source" library to prevent multiple metadata entries for the same product.

ANNEX C: Test and Certification

C.1. NSILI Server Test Suite

The development of an NSILI Server Test Suite is an ongoing collaboration between NCIA, MAJIC and NSILI CST members. The purpose of the test suite is twofold. First, it provides ability to validate STANAG 4559 compliance against the agreed baseline (currently STANAG 4559, Edition 3). Second, it serves to identify change requirements to ensure the STANAG and/or AEDP-5 documentation remains up to date. In addition to testing, the test suite and its resident tools can be used to support set-up and execution for exercises, demonstrations, and even operational venues.

As stated above, the development of the Test Suite is ongoing. In its initial stages the test suite is designed to support MAJIC2 agreements that are not part of the STANAG 4559. Use of the Test Suite without modification could result in an external (to MAJIC2) server implementation being considered (partially) "red" by the Test Suite even though it completely complies with STANAG4559, Edition 3. Future development should address MAJIC2 variances and be integrated with development of the next edition of STANAG 4559 to ensure interoperability between "MAJIC2 CSDs" and "external CSDs".

Another consideration is the types of testing. CSD server testing could consist of two distinct types of tests, CSD to CSD interoperability testing and "end-to-end" testing. End-to-end tests have multiple steps such as 1) ingestion of an JISR product into the CSD, 2) alerting by CSD of subscribing users, 3) synchronisation on metadata to other CSD servers, 4) searching on local and remote CSD for products by users, 5) retrieval of product from a remote CSD to a local CSD server and 6) retrieval of products from a local CSD server to a user application.

Determining a baseline is a core test requirement, because there will almost always be multiple implementations of any STANAG Edition. To ensure a valid test of Standards conformance the Custodian and the CST must first agree to the baseline, and more specifically, agree to a base line data model. Until recently data models were incorporated into the STANAG documents; however, under current procedures the STANAG will not include technical information. The current documentation strategy for NSILI is to post NSILI data models either on the NMRR or on the US Metadata Registry (MDR), whichever is accessible to NATO users at the time. Implementers will be able to use the data models as they choose. The source of the data models may be from experimentation activities, such as those conducted by NCIA, or from models developed within an operational environment. Currently there are sample data models that may be viewed once access is approved by navigating to the NMRR site: <https://nmrr.nc3a.nato.int> and using the following sequence:

- After login, in the menu, click "Browse > Documents".
- Then, under "By Namespace" click on "NATO interim > NIIA > STANAG 4559". (A small note: please, wait until the page is fully loaded before clicking any of the links, otherwise the next page will display incorrectly).
- In the main window, you'll see the list of registered documents. Click the one you would like to review.
- At the bottom you'll see the document's registration details, including a link to view and download the artifact respectively (i.e. "View artifact" and "Download artifact").
- Before you can view/download you may be asked to renew your log in information.

C.2. Test Venues:

The STANAG 4559 Custodian seeks venues for testing that will enable those Nations with STANAG 4559 compliant library systems to investigate their respective levels of interoperability within Coalition networks. Past venues have included Trials (Unified Vision), lab tests and demonstrations (Empire/Enterprise Challenge for example).

C.3. Test Preconditions:

The current version of the NSILI Server Test Suite requires data to be present in the server being tested otherwise a number of tests will fail while others will fail to fully test the server in a meaningful way. Certain tests may also fail if the data is in the wrong format, notably the testGet_related_files test case would fail if there is no data containing related THUMBNAIL files. This can be resolved by ensuring a standard data-set is loaded into the Server before the test suite is run. The exact makeup of a data-set needed to fully exercise the Test Suite is outside of the remit of this document but the following provides some guidelines:

The dataset should conform to the following rules

- It should contain known quantities of products of a variety of all data types supported by the STANAG
- It should contain a known number of products within a specified geographic bounds
- It should contain a known number of products with associated related files and their types

If the STANAG Test Suite specification is updated to define such a data set then the Test Suite could be updated to provide more checking of return types. For example given, an assumption about the dataset contents, the ProductManager.get_related_file_types() method could be checked to ensure that the correct range of types for a given product type are supported.

The Test Suite specification should *not* attempt to define the exact contents of a data set; instead it should merely define a set of rules that ensures the Test Suite can properly check that the Server being tested conforms to the STANAG.

C.4. STANAG Compliance:

Compliance to the ISR Standardization Agreements and the NIIA are not rigidly qualified for all of the STANAGs, although initiatives are under consideration that will establish test and certification laboratories or test suites for each. The NIIA provides a foundation and means to collect, store and exchange data and adheres to accomplishing the levels of interoperability. For these reasons the Joint ISR Capability Group leads or participates in trial, exercise and experiment events that help implementers determine their levels of interoperability and development program to accomplish the desired exchange capability.

NATO interoperability publications define four levels of interoperability. Those levels of interoperability are:

- Degree 1 (Unstructured Data Exchange): Involves the exchange of human-interpretable

unstructured data such as the free text found in operational estimates, analysis and papers.

Degree 2 (Structured Data Exchange): Involves the exchange of human-interpretable structured data intended for manual and/or automated handling, but requires manual compilation, receipt and/or message dispatch.

Degree 3 (Seamless Sharing of Data): Involves the automated sharing of data amongst systems based on a common exchange model.

Degree 4 (Seamless Sharing of Information): An extension of degree 3 to the universal interpretation of information through data processing based on co-operating applications.

ANNEX D: Configuration Management

CONFIGURATION MANAGEMENT FOR THE STANAG 4559 NATO STANDARD ISR LIBRARY INTERFACE

D.1. Purpose

The purpose of this Annex is to provide the framework for the management of STANAG 4559 and all associated documents. If the guidance discussed below conflicts with guidance contained in the AAP-3 version that is current at the time, the AAP-3 guidance will take precedence.

Documents related to Configuration Management:

STANAG 4559

- Allied Engineering Document Publication – 5, NATO Standard ISR Library Interface Implementation Guidance (AEDP-5).
- NSILI Server Test Suite Specification Documentation. A new set of Coalition Shared Data (CSD) system test tools is in development by the MAJIIC2 Program, which is supported by NCIA. These test tools require maturation on a continuous basis with support from the MAJIIC2 Nations and other participating NATO Nations. The test tools currently reside on NCIA's Collaborative Development Test Evaluation (CDTE) network, which requires implementers to coordinate with NCIA for access. Other access paths are being considered, but are not available at this writing.

Other Referenced Documents

- NATO AEDP-2, NATO Intelligence Integration Architecture (NIIA).
- AAP-3 *Procedures for the Development, Preparation, Production, and the Updating of NATO Standardization Agreements (STANAGs) and Allied Publications (APs)*
- AC/224(JCGISR)D(2008)0003; *Terms of Reference (TOR) for the NATO NAFAG Joint Capability Group on Intelligence, Surveillance, and Reconnaissance (JCGISR) and the All-Source Intelligence integration Sub-Group (ASIISG)*

D.2. Scope

This annex provides the framework for configuration management of STANAG 4559 and all associated documents. Participating NATO member nations define their respective levels of participation and all NATO member nations have equal opportunity to have their respective positions voiced in the STANAG 4559 community. Decisions made within this framework are subject to final approval of NATO Air Forces Armament Group (NAFAG) Joint Capabilities Group on ISR (JCGISR,) in order to ensure the proper placement of STANAG

4559 within the overall NATO Imagery Interoperability Architecture (NIIA). The configuration management structure outlined in this document is compatible with the NATO guidelines defined in AAP-3, Procedures for the Development, Preparation, Production, and the Updating of NATO Standardization Agreements (STANAGs) and Allied Publications (APs).

D.3. STANAG Management Organization

The STANAG 4559 is managed by an organizational structure that includes the Joint Capability Group for ISR (JCGISR) as the Functional Manager of the STANAG. Technical application of the STANAG within the NIIA is managed by the All Source Intelligence Integration Sub-Working Group (ASIISG), which nominates the Custodian who after appointment by the JCGISR, will maintain and coordinate the technical content of the document. The JCGISR also supports the Custodian Support Team by supplying National representatives and technical experts.

The NAFAG JCGISR will:

- Appoint a custodian for STANAG 4559
- Resolve conflicts between Nations on the disposition of proposed changes to STANAG 4559 when conflicts cannot be jointly resolved by the Custodian and the National Points of Contact (POCs); prepare a Decision Sheet; and provide notification to the Custodian.
- Assess all proposed amendments or new editions of STANAG 4559 to determine the impact of included changes on other STANAGs for which JCGISR is responsible.
- Approve amendments or new editions to STANAG 4559 prior to forwarding them to the nations for ratification or to the Chairman, NATO Standards Agency (NSA) for promulgation.

The NAFAG JCGISR/All-Source Intelligence Integration Sub- Group (ASIISG) will:

- Nominate a Custodian for STANAG 4559 for appointment by the JCGISR.
- Resolve technical conflicts in implementations of STANAG 4559 and its relation to the NIIA.
- Provide technical guidance to the Custodian based on the interests of the national POC's and the NIIA.
- Provide a forum for technical exchange of information relating to implementations of the NIIA and interoperability of the STANAG for which the JCGISR is responsible.
- Provide recommendations to the JCGISR based on the best policies for the NIIA as agreed by the ASIISG.
- Organize and conduct validation trials and experiments that validate the implementation and interoperability of the STANAG's for which the JCGISR is responsible e.g. the NIIA.

The STANAG Custodian: The STANAG 4559 custodian is responsible to the JCGISR for all STANAG 4559 activity. The Custodian is the only individual to receive tasking from and report to the ASIISG for the JCGISR on STANAG 4559. This authority can be delegated to other members of the STANAG 4559 community, but responsibility for the tasking and reporting resides with the Custodian. Specific duties include, but are not limited to:

- Maintaining a list of the National POCs and provides the list to the JCGISR Secretary for

- publication on the NAFAG internet web site.
- Maintaining a list of POCs in organizations /agencies responsible for implementing STANAG 4559 in national systems.
- Receiving proposed changes to STANAG 4559 from National POCs.
- Tracking proposed changes and maintains current status of each proposed change until final disposition.
- Reviewing proposed changes to determine if the proposed changes are substantive or editorial
- Determining if the circumstances of a proposed substantive change dictate the use of “Fast Track” processing as described in “Proposal for the “Fast Track” Processing of the Interoperability STANAGs.”
- Determining the disposition of proposed editorial changes.
- Forwarding a copy of each substantive proposed change to national POCs (and implementation POCs when appropriate) for review and recommendations.
- Calling for and presiding over meetings of national POCs when required.
- Coordinating with the Custodians of other STANAGs sponsored by JCGISR.
- Providing STANAG amendments or new editions to JCGISR for approval through the ASIISG.
- Reporting to the JCGISR/ASIISG on status of STANAG 4559 and proposed changes.
- Coordinating with the custodians of the USIGS Geospatial and Imagery Access Services Specification (GIAS) and USIGS Common Object Specification (UCOS) referenced in STANAG 4559 to assure that changes to those documents are reviewed for applicability to STANAG 4559 and communicate JCGISR/ASIISG concerns to those custodians.
- Coordinating with liaison organizations that utilize and are impacted by, or identify modifications to STANAG 4559, e.g. Multi-sensor Airborne/ground Joint ISR Interoperability Coalition (MAJIIC) and the Digital Geospatial Information Working Group (DGIWG).

The STANAG 4559 POC:

Each NATO member nation’s JCGISR Representative can appoint a National POC for STANAG 4559 by providing the name, organization, address, telephone and telefax numbers, and electronic mail address of their 4559 POC to the STANAG 4559 Custodian. The National POC for STANAG 4559 can be from government or industry as chosen by the JCGISR representative. The national POC for STANAG 4559 is the official spokesman for all participants from that nation. The authority of the national POC can be delegated to another individual from that nation if the POC is temporarily unable to function. The delegation shall be in writing to the Custodian. The substitute shall have all authority and responsibility of the regular national POC. Specific duties of the National POCs include, but are not limited to:

- Review and evaluate proposed changes to STANAG 4559 or supporting documents submitted by interested parties within their respective nations.
- Forward proposed changes and recommendations for “Fast Track” processing to the Custodian for action.
- Establish procedures for developing their respective national position on proposed changes to STANAG 4559. These procedures can use whatever process is appropriate to that nation, but ultimately the national representative will voice the official national position to the 4559 Custodian.
- When required, determine if implementing organizations/agencies within their respective Nations will participate in “Fast Track” processing of substantive changes to STANAG 4559, and will identify implementation POCs to the Custodian.

- Distribute proposed changes received from the custodian to interested persons or organizations within their nation and report their Nation's position on proposed changes to the Custodian.
- Represent their Nation at meetings of the National POCs when such meetings are called by the Custodian.

NATO JCGISR/ASIISG Secretary: The JCGISR Secretary is responsible for maintaining the JCGISR content on the NATO web pages on which STANAG 4559 is posted and other NAFAG controlled websites. The JCGISR Secretary supports STANAG 4559 configuration management by:

- Posting approved revisions of STANAG 4559 to the internet pages within 45 days of the JCGISR/ASIISG meeting, unless other arrangements are agreed during the meeting.
- Maintain a list of the national POCs for STANAG 4559 on the web page.
- Distribute amendments or new editions containing substantive changes to the Nations for ratification in accordance with AAP-3.
- Submit ratified amendments or new editions of the STANAG or amendments or new editions containing only editorial changes to the Chairman NSA for promulgation.

D.4. Change Management

Note: The management process for NATO documentation is undergoing change and some of the processes described below may change in the future; however, this AEDP will be processed using the below procedures. For current change management procedures that will apply to new documentation, refer to the most current version of the AAP-03. Also, refer to Revision updates to AAP-32 as appropriate.

Two processes are employed for management of changes to existing STANAGs such as STANAG 4559. Both processes are illustrated in Figure 4. The first process deals with urgent changes, which follow a faster process using simultaneous reviews at different coordination levels; the process is illustrated in the left column of Figure 4. The second process deals with routine changes, which are processed as illustrated in the right column of Figure 4.

Any interested individual or organization may submit a request to change the content or structure of STANAG 4559 at any time. Individuals or organizations from nations that have identified a National POC for STANAG 4559 will submit proposed changes to their National POC. Individuals or organizations from NATO nations that have not identified a National POC for STANAG 4559 will submit proposed changes to their nation's JCGISR representative.

- National POCs will evaluate proposed changes. If the National POC endorses the proposed change, the National POC forwards the proposed change to the Custodian using the form illustrated in Figure 5 of this document. If the National POC does not endorse the proposed change, the National POC returns the proposed change to the originator.
- Change requests shall contain the information reflected in Appendix A and may be submitted to the Custodian via mail, electronic mail, or telefax. Changes submitted as a result of ambiguities or problems discovered during implementation of STANAG 4559 must provide a proposed solution to the problem encountered and an assessment of the urgency and impact to other implementations.

- The Custodian logs the change into the configuration management system, acknowledges receipt of the change, and reviews the proposed change to determine if the change is substantive or editorial.
- If the proposed change is editorial, the Custodian will determine the appropriate disposition of the change without input from the national POCs. Approved changes will be incorporated into the next forthcoming amendment or edition of STANAG 4559.
- If the proposed change is substantive, the custodian disseminates the proposed change simultaneously to the national POCs, implementation POCs, and to the Custodians of associated STANAGs for their review and comment. When the Custodian disseminates a change in this manner the Custodian will establish a suspense date for replies from the POCs. **POCs who do not submit replies by the suspense date will be deemed to agree to the proposed change.**
- If there is no disagreement on the proposed change, the custodian notifies the POCs that the change will be adopted and incorporates the change into the ratification draft of the next amendment or new edition of the STANAG. The change will be documented on the Errata Sheet and the Errata Sheet will be posted on the internet with the promulgated STANAG for the duration of the ratification process for the new amendment or edition.
- Normal procedures: If the custodian determines that there is no need to expedite processing of a particular change proposal, the custodian will compile proposed changes on an Errata Sheet, which will be updated and reposted on the internet with the promulgated edition of the STANAG. The Errata Sheet will continue to be updated and reposted with new changes until the Custodian and POC's support submission of an amendment or new edition of the STANAG. Compiled changes will be documented and decisions recorded using the Change Matrix shown In Figure 6 of this Annex. National POCs will be notified of an updated Errata Sheet through STANAG 4559 Custodial Support Team meetings and through status reports to the ASIISG.
- National POCs disseminate proposed change(s) to interested individuals or organizations within their respective nations. National POCs and other interested parties within their nations assess the impact of the proposed changes and develop a national position on each change using procedures established by the National POC. National POCs then report their nation's position to the Custodian.
- If there is no disagreement on the proposed change, the custodian incorporates the change into a new version of the Errata Sheet and posts the Errata Sheet on the internet at the same location as the promulgated STANAG. The change will then be incorporated into the ratification draft of the next amendment or new edition of the STANAG.
- If National positions on a proposed change conflict, the Custodian will call a meeting of the national POCs to discuss the proposed change and attempt to reach consensus. Procedures for announcing and conducting meetings of the National POCs are described in paragraph 5.
- When National POCs cannot reach agreement on a proposed change, the Custodian forwards the proposed change, a memo for record outlining the various National

positions on the change, and the Custodian's recommendations to ASIISG where the disposition of the change will be decided. ASIISG decisions will be documented via an ASIISG Decision Sheet. The ASIISG will report its recommendations to the JCGISR for coordination purposes; it is expected that recommendations made by the ASIISG will be endorsed by the JCGISR.

- The Custodian will forward all proposed amendments or new editions of the STANAG to the ASIISG for endorsement. The ASIISG will recommend the proposed amendments or new editions to the JCGISR for ratification.
- The JCGISR will approve amendments or new editions of the STANAG that contain only editorial changes based on the endorsement of the ASIISG. These approvals will be conducted in JCGISR plenary to expedite submission to the NSA Chairman for promulgation.
- The ASIISG will approve amendments or new editions containing substantive changes prior to recommendation to the JCGISR for ratification.
- The JCGISR will approve amendments or new editions containing substantive changes, based on the endorsement of the ASIISG prior to distribution to the Nations for ratification.

Figure 4 (next page) diagrams the change management process.

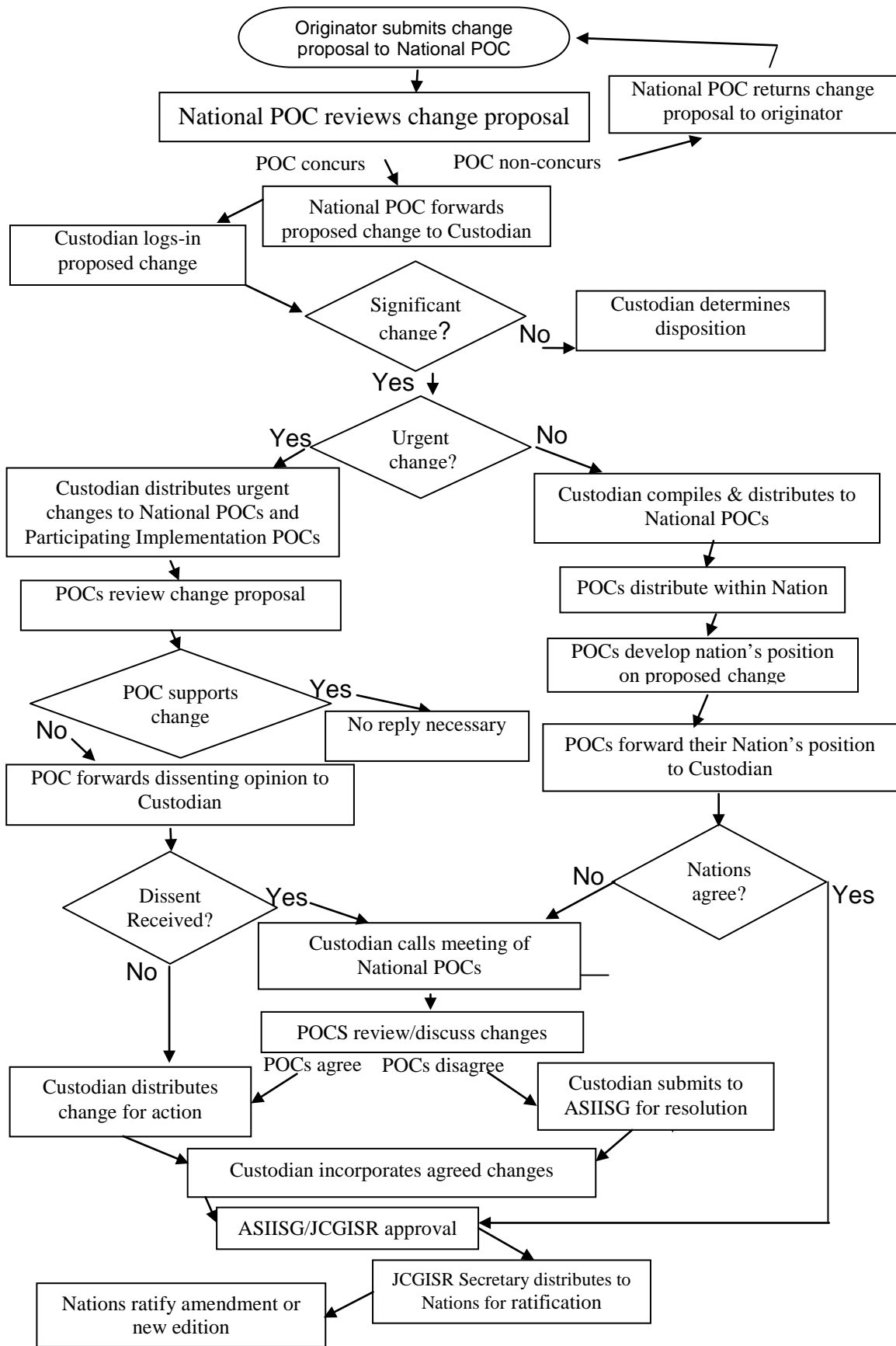


Figure 4 – Change Management Process

Figure 5 (next page) is the Change Proposal Form.

CHANGE PROPOSAL FORM

STANDARDIZATION DOCUMENT CHANGE PROPOSAL

INSTRUCTIONS

1. Change proposals may be submitted on this form through either mail or telefax, or by electronic mail following the same order and content as this form.
2. Originator completes sections 1-16.
3. Originator forwards to the respective national representative. National representative is official representative to the representatives to the NATO Joint Capability Group for ISR (JCGISR). (See the NATO Defense Investment Division Web Portal <https://diweb.hq.nato.int>.)
4. National representative approves or rejects proposal from their nation by completing sections 17-25.
 - Approved proposals are forwarded to the Custodian for STANAG 4559 NATO Standard ISR Library Interface and Allied Engineering Documentation Publication-5.
 - Rejected proposals are annotated with the reason for disapproval and returned to the originator.

Note: This form may be used to submit changes to any document related to STANAG 4559 and AEDP-5. This form may not be used to request copies of these documents. The documents are available on the NATO Standardization Agency catalog nsa.nato.int or through national mission distribution channels for NATO documents.

RECOMMENDED CHANGE: (continue on additional sheets as necessary) page 1 of 1

1. Document Number: <div style="text-align: center; font-size: 1.2em;">Choose an item.</div>	2. Document Version/Release Number: <div style="text-align: center; font-size: 1.2em;">Choose an item.</div>	3. Document Date: <div style="text-align: center; font-size: 1.2em;">Choose an item.</div>
---	---	---

4. Document Title: <div style="text-align: center; font-size: 1.2em;">Choose an item.</div>	
5. Proposed Change to: (Section, Paragraph, Line, Page)	6. Change Class: I <input type="checkbox"/> II <input type="checkbox"/>

7. Current Wording: <div style="font-size: 1.2em;">See attached.</div>	8. Proposed Wording:
---	----------------------

9. Reason/Rationale:

10. Originator's Name:	13. Originator's Telephone Number:
11. Originator's Organization:	14. Originator's Telefax Number:
12. Originator's Mailing Address:	15. Originator's Email Address:
	16. Date Submitted:
17. Nat'l Rep Name:	20. Nat'l Rep Telephone Number: +
18. Nat'l Rep Organization:	21. Nat'l Rep Telefax Number:
19. Nat'l Rep Mailing Address:	22. Nat'l Rep Email Address:
	23. Date of Approval/Rejection:

24. Change Proposal: Approved Rejected

25. Rejection Rationale:

<p>Submit proposals to: STANAG 4559 Custodian, NGA OCIOCE L-66, 3838 VOGEL ROAD, ARNOLD MO 63010-6238 USA Telefax+1 314 676 3015 Email: Laura.A.Moore@NGA.Mil</p>	<p>26. Date Logged by STANAG 4559 and AEDP-5 Custodian/initials:</p> <div style="text-align: center; margin-top: 20px;"> <hr style="width: 80%; margin: 0 auto;"/> Laura Moore Custodian, STANAG 4559 and AEDP-5 </div>
---	--

Figure 5 – Change Proposal Form

The Consolidated Change Proposal report and log format is illustrated in figure 6 below:

**SAMPLE CONSOLIDATED CHANGE PROPOSAL REPORT AND LOG
FORMAT**

Proposed Changes to
NATO AEDP-5, Edition 2
Date of Issue:XXXX _____

CR #	Accept or Reject	Submitter	Submission Date or Event	Change Add Delete	Type of Change	Document Location	Current Text	Recommended Text	Rationale	Date Complete
12-01		Mueller	Feb 11	Delete	S	Pg. 8, Annex A, Sec. 1.3, Table A-X-ORB	ORB Table	Delete Table	Not needed as there is only 1 language currently used in developing new systems	6 Feb 12
12-02		IOSB	Jan 12	delete	S	Pg. 23, Annex A, Sec. 2.8.2.12	New model does not permit us of spaces between entity names	Delete para 1 and 2 sub bullets under it and renum next para. Also added explanatory note about Oracle g11 (current ver.)	Updates to new model rules	6 Feb 12
12-03		IOSB	Jan 12	Delete	S	Pg. 28, Annex A, Sec. 2.8.4.1	Same as para 2.8.2.12 above.	Delete para 1 and 2 sub bullets under it and renum next para. Also added explanatory note about Oracle g11 (current ver.)	Updates to new model rules	6 Feb 12
12-04		IOSB	Jan 12	Delete and add new	S	Apply to whole document	HTTPS throughout	Change HTTPS to HTTP(S) throughout document	Correctly describes use of either HTTP or HTTPS	9 Jan 12
12-05		IOSB	Jan 12	Delete	S	Pg.37, Annex A, Sec. 2.9.4.1	Para describes CORE attribute	Delete Para beginning with "CORE: request all attributes..."	CORE is not a supported attribute in the new model	
12-06		Mueller	Feb 11	Delete	S	Pg. 37, Annex A, Sec. 2.9.4.2	Includes DARFI and NSIL_CORE in discussion on ordering related files	Delete ref to DARFI change NSIL_CORE to NSIL_ALL_VIEW	Both terms are no longer supported in the STANAG	Jan 12
12-07		IOSB	Jan 12	Delete and add new	S	Pg. 37, Annex A, Sec. 2.9.4.2, 4 th Sub Para.	Current wording is confusing and may be incorrect	Delete current 4 th Sub Para and replace with, "all (related) file access should be performed by accessing the URLs that are given in the metadata."	Simplifies the related files method.	Feb 12

Figure 6 – Consolidated Change Proposal Report and Log Format Example

D.5. Meeting Procedures

- All meetings will be announced with a minimum of 60 days notice.

- All meetings will be conducted in English. Those nations requiring the materials in different languages are responsible for translating the materials. Attendees to the meetings should be proficient enough in English to contribute to the meeting in English.
- National POCs may invite other individuals from their nations to participate during the meeting. These additional participants may be government or contractor personnel. The intent of having additional personnel participate is to provide technical, operational, or procedural expertise that may not be resident with the National POCs and to allow participation by those who are developing systems using STANAG 4559.
- During the meeting, the custodian will direct discussion of each proposed change and will attempt to resolve any areas of disagreement.
- If the National POCs are unable to resolve their disagreements the group will prepare a memo for record outlining the proposed change and the various National positions. The Custodian will provide this memo for record to ASIISG when referring the matter to ASIISG for a decision.
- Every effort will be made to determine the disposition of all proposed changes during the meeting. However, a decision may be deferred if the Custodian determines that additional investigation/review is required. In such cases, the Custodian will assign responsibility for additional study/review.
- Minutes of all meetings will be distributed within 14 days of the completion of the meeting. The minutes will include a record to document approved and disapproved changes, identify the status of all outstanding changes, and identify issues to be taken forward to ASIISG.

PAGE INTENTIONALLY LEFT BLANK

ANNEX E: Data Models and Metadata

New data models will be required as NSILI implementations are adapted to the suite of ISR data formats. The Metadata Harmonization TST will provide the attributes that are required to describe a data type. As an implementation begins to provide for a particular dataset, the model should be developed in coordination with the NSILI CST, the Custodian of the dataset, the Metadata Harmonization TST, and finally the ASIISG. Data models will provide an xml schema if appropriate. Upon approval by the NSILI CST and appropriate development bodies, the Custodian will incorporate the model into the STANAG 4559 Errata Sheet where it will be posted with the STANAG on the NAFAG web pages. An alternative plan under consideration is to post future data models separately on the web site when it becomes available, and install a link/reference to the data model web location in the Errata Sheet and/or document. Currently the web site: <https://nc3a.nmrr.nato.int> may be reached once permissions have been obtained. To locate STANAG 4559 draft documentation at the web site use the following steps:

- After login, in the menu, click “Browse > Documents”.
- Then, under “By Namespace” click on “NATO interim > NIIA > STANAG 4559”. (A small note: please, wait until the page is fully loaded before clicking any of the links, otherwise the next page will display incorrectly).
- In the main window, you’ll see the list of registered documents. Click the one you would like to review.
- At the bottom you’ll see the document’s registration details, including a link to view and download the artifact respectively (i.e. “View artifact” and “Download artifact”).
- Before you can view/download you may be asked to renew your log in information.

ANNEX F: Employment Guidance

F.1. Video and Streaming Data

The NSIL_STREAM entity is part of the minimum layer for 4559 Edition 3 and is a metadata only entity, i.e. there are no physical products related to this entity stored within the 4559 IPL. The NSIL_STREAM entity being a metadata only entity allows the source provider to update the attributes, as necessary, to reflect the changing characteristic of the stream. The NSIL_STREAM entity provides a 4559 client with sufficient information for a client to discover and be able to acquire a product stream produced from an external data source or to access a file server streaming a product file to the requesting client, as for example a video clipping server streaming multiple video clips.

The NSIL_STREAM has been enumerated to deliver products formatted as STANAG 4609 (video), STANAG 4607 (GMTI), and STANAG 5516 (TDL) over IP networks. It should be noted, for the current release of STANAG 4559, TDL is encoded with the NATO AGS Capability Testbed (NACT) header, other TDL IP streams (JREAP) is not currently supported in the NSIL_STREAM.standard enumeration. The NSIL_STREAM metadata should provide sufficient information to provide the 4559 client the ability to query and retrieve the stream. The metadata attribute information and time period for the stream is the responsibility of the stream source provider. The NSIL_STREAM entity information may be augmented by completing other entity attributes, as provided in the STANAG, possible entities could include NSIL_Coverage, NSIL_PART/NSIL_VIDEO.

Clients should note that the NSIL_STREAM is an advertisement of available products available as a network stream (UDP Broadcast, Multicast, etc) only. Access to any NSIL_STREAM results by clients is dependent upon the system publishing the data and the network having proper support for transmitting the content. Reasons clients cannot access a given NSIL_STREAM entry are varied, but the most common include enclaves not routing multicast, Network Address Translation (NAT) due to network devices and/or cross domain devices requiring IP address mappings. Systems publishing NSIL_STREAM entries need to work with the relevant network and systems personnel to determine how the entry will be made available to possible clients.

F.2. Removing Data from a CSD

When an artifact in the library is no longer operationally required, it is usually marked as "obsolete" by a client invoking the "delete()" operation on the UpdateMgr. This operation does not cause the artifact to completely vanish from the library - instead the "NSILCARD.status" field will change to "OBSOLETE". In addition, some library implementations also make the product file and related files unavailable (removing URL

from metadata, making the file inaccessible via an HTTP server). This additional technique is optional and should not be relied upon.

Removing, in terms of the artifact completely vanishing from the library (both metadata and files), is not supported via the STANAG4559 interface. Removing files/data must be performed through an additional administrative interface that is not described by STANAG 4559, either in terms of functionality or interface. When coordinating the removal process, special attention needs to be given to the replication and synchronization capability of the different library implementations, as a query based replication mechanism is usually employed. Such a mechanism makes it impossible to detect vanished artifacts in order to cascade the removal to other libraries.

When an artifact is operationally relevant, but is not required to be immediately available online, an archiving process is usually performed (generally to save storage space for large files). As for removal, it is not possible to perform the removal action using the STANAG 4559 interface archiving process. Removal can only be done with additional capabilities integrated into a library implementation. Also the rules on how archived artifacts need to be stated in the metadata are not defined (though some fields exist that could be used in such a process).

As the processes of archiving and removal are up to now not standardized, performing such action requires previous coordination between all participants. This coordination usually needs to take place prior to fielding as this coordination usually causes software changes to be done for the various libraries.

F.3. United States Imagery and Geospatial Information Systems (USIGS) Architecture Common Object Specification (UCOS) Interface Definition Language (IDL)

To help developers incorporate the GIAS/UCOS IDL specification this appendix provides a high level definitions and descriptions of the interfaces and structures. The reference details and IDL specifications are maintained in the GIAS reference document and the accompanying UCOS Common Object Specification. Both the GIAS and the UCOS documents can be found at:

<http://www.nato.int/structur/AC/224/standard/4559/4559.htm>

To ensure interoperability, all systems that must interoperate must make the same interpretations concerning the specifications contained within the reference documents. The GIAS and UCOS reference documents provide a general specification, and a profile specification for the intended area of use. They are a critical supplement to these specifications and describes the specific interpretations, limits and conventions to be used within an area of use. This STANAG is considered a *profile* of those specifications and is specific to its use within the ISR arena.

All elements of the GIAS definition are contained in the GIAS module, which identifies and defines data types, interfaces, operations, and exceptions. The interfaces defined in GIAS use the exception model defined in USIGS Common Object Services (UCOS) Specification. That specification defines a general purpose model which the GIAS specification extends by defining a set of error condition identifiers.

The GIAS interfaces are partitioned into four activity categories: library; request managers; request objects; and callback/product objects. The general notion is that a GIAS client requires access to a Library, which

is accessible through the GIAS interfaces. The GIAS client interacts with the Library to select and request access to a manager of a specific type. (“manager selection activity category”). Using the provided Manager the client can submit requests for the Library to perform tasks (“request submittal activity category”). Each request submittal returns a Request object. The GIAS client then uses the Request object to monitor progress on the task and to retrieve the results. The Request object also provides a mechanism (a Callback) to allow a client to be notified of the progress of the task. The GIAS client can also obtain information (“ ancestor information activity category”) on a specific request or manager. This allows a GIAS client to determine for any Request the Manager that is managing it, and for any Manager, determine the Library(s) it services.

F.4. CSD Replication Mechanism

If the STANAG4559 only pertains to the interface to the client, the benefit of CSD servers being able to provide data across the whole enterprise is lost. Metadata is replicated to achieve a consistent state between the participating servers, in other words, the servers become synchronized. If the mechanism by which the metadata and the files are replicated across the servers is not described, then the individual vendors that use the CSD 4559 server interface will develop their individual mechanisms, making synchronization problematic, if not impossible. This is because each time the servers are connected together, it will be necessary for the vendors to modify their source code result in order to be able to synchronize. Not addressing this issue up front creates the potential for chaos to ensure when a CSD network is constructed in a deployed environment.

There are cases where the STANAG4559 interface is used to provide a query interface for sensors. If the idea is to make the data on the sensor available for a small subset of the enterprise (e.g. the ground station), then many would agree that it is not necessary for them to provide a replication mechanism for that instance. If, however, the idea is to make all the data available for the whole enterprise, then even such sensor systems should be incorporated into a replication infrastructure that could be set up for a read only capability where it would retrieve data from other CSD servers.