**DRAFT ISA Standard**

**DRAFT**

**ISA-95.00.07**

**Enterprise/Control System Integration – Alias Service Model**

**FOR USE AND REVIEW ONLY BY MEMBERS OF THE ISA95 COMMITTEE:**

ISA-95.00.07–xxxx  
Alias Service Model

ISBN: xxx.xxx.xxxx

Copyright © 2015 by ISA. All rights reserved. Not for resale. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), without the prior written permission of the Publisher.

ISA  
67 Alexander Drive  
P. O. Box 12277  
Research Triangle Park, North Carolina 27709  
USA

CONTENTS

FOREWORD 5

INTRODUCTION 6

1 Scope 8

2 References 8

3 Definitions, abbreviations, and conventions 8

3.1 Terms and definitions 8

3.2 Abbreviations 9

3.3 Conventions 9

4 Identifying objects and enumerations in data exchanges 9

4.1 Converting data element IDs 9

4.2 Principle item 10

4.3 Identifier categories 10

4.4 Alias directory use 10

5 Alias directory object model 11

5.1 Alias directory model 11

5.2 Alias directory elements 12

5.3 Category 13

5.4 Principle item 14

5.5 Principle item property 14

5.6 Namespace 15

5.7 Namespace item 15

5.8 Namespace item property 16

6 Alias services 17

6.1 Alias service elements 17

6.2 Security 17

6.3 ASM service type definitions 19

6.4 ASM service returns and faults 19

6.5 ASM mapping services 19

7 Alias directory model transactions 26

7.1 Alias directory elements 26

7.2 Alias directory verbs 27

7.3 Alias directory verb actions 27

8 Completeness, compliance and conformance 28

8.1 Completeness 28

8.2 Compliance 28

8.3 Conformance 28

Annex A – (Normative) Recommended value properties 30

1. ParentIdentifierID 30

2. ChildIdentifierID 30

3. PossibleEquivalentIdentifierID 30

4. EffectiveStartDate 30

5. EffectiveEndDate 30

Annex B – (Normative) Recommended categories 31

Annex C – Use scenarios 36

Annex D – FAQ 42

Figures

Figure 1 – Steps in application-to-application communication 7

Figure 2 – Specifications in application-to-application communication 11

Figure 3 – Alias directory model 12

Figure 4 – Security of identifier registries 18

Figure 5 – Object grouping for the alias directory model 27

Figure 6 – ASM simple example using services 36

Figure 7 – ASM simple example using messages 37

Figure 8 – ASM cache example using services 38

Figure 9 – ASM cache example using messages 39

Figure 10 – ASM multiple methods 40

Tables

[Table 1 – Attributes of alias directory 13](#_Toc469647283)

[Table 2 – Attributes of category 13](#_Toc469647284)

[Table 3 – Attributes of principle item 14](#_Toc469647285)

[Table 4 – Attributes of principle item property 14](#_Toc469647286)

[Table 5 – Attributes of namespace 15](#_Toc469647287)

[Table 6 – Attributes of namespace item 16](#_Toc469647288)

[Table 7 – Attributes of namespace item property 17](#_Toc469647289)

[Table 8 – ASM type definitions 19](#_Toc469647290)

[Table 9 – ASM service returns and fault definitions 19](#_Toc469647291)

[Table 10 – Get namespace IDs 20](#_Toc469647292)

[Table 11 – Get principle IDs 21](#_Toc469647293)

[Table 12 – Get equivalent IDs 22](#_Toc469647294)

[Table 13 – Get alias directory information 23](#_Toc469647295)

[Table 14 – Get alias directory information rules 23](#_Toc469647296)

[Table 15 – Set namespace items 24](#_Toc469647297)

[Table 16 – Set principle items 25](#_Toc469647298)

[Table 17 – Set equivalent identifiers 26](#_Toc469647299)

[Table 18 – Alias directory verb actions 27](#_Toc469647300)

[Table 19 – Supported services and verb/noun transactions 29](#_Toc469647301)

[Table 20 – Standard category names 31](#_Toc469647302)

[Table 21 - Object class and property categories 33](#_Toc469647303)

**Enterprise-Control System Integration**

**Part 7: Alias Service Model**

# FOREWORD

ISA95 is a multi-part standard that defines enterprise to control system integration. This document is Part 7 that defines an alias service model (ASM) that includes a set of services and exchange information used to map identifiers from one application’s or system’s identifier repository to another application’s or system’s identifier repository.

Clause 4 is normative. It defines the general service model and functions of the ASM services.

Clause 5 is normative. It defines the ASM object model.

Clause 6 is normative. It defines the ASM services.

Clause 7 is normative. It defines the ASM transactions.

Clause 8 is normative. It defines conformance points for compliance.

Annex A is normative. It provides value property values.

Annex B is normative. It provides category names.

Annex C is informative. It provides example of use of the ASM.

Annex D is informative. It provides answers to frequently asked questions.

As currently envisioned, ANSI/ISA95 consists of the following parts under the general title Enterprise-Control System Integration:

- Part 1: Models and terminology

- Part 2: Objects and attributes for enterprise-control system Integration

- Part 3: Activity models of manufacturing operations management

- Part 4: Object models and attributes of manufacturing operations management

- Part 5: Business to manufacturing transactions

- Part 6: Messaging service model (MSM)

- Part 7: Alias service model (ASM)

# INTRODUCTION

This part of the standard is based on the use of ISA-95 object models defined in Parts 2, 4 and 5 (Parts 1 and 3 do not contain object models) to define a set of services and messages that may be used to facilitate the mapping of identifiable elements between communicating applications. Applications may have, and often do have, different identifications for the same objects. In any exchange all parties to the exchange must agree on the names or identifiers of the objects. Often this agreement is managed through a set of alias identification registration services. This part defines an alias service model (ASM) and formal object model for translating identifers in Level 3 to Level 4 and Level 3 to Level 3 exchanges. This part defines a simple implementation approach to a subset of the much broader issue that is addressed by ISO/IEC 11179 Information Technology – Metadata Registries (MDR) and ISO/IEC 19763 Information Technology – Metamodel framework for interoperability (MFI).

All references to levels in this standard pertain to the levels of the function hierarchy model identified in Part 1 of this standard.

This part of the standard defines a set of services definitions that are designed to provide the functionality needed for a vendor independent method for translating identifiers across different namespaces. Using a common representation of objects, such as enumeration strings, property names, and equipment tag names, is an important element of any integrated system. Enterprise/Control system integration involves multiple different steps to exchange data between different computer system applications, as shown in Figure 1.

1. The applications usually have different internal representations of exchanged objects in their own local data stores. In an exchange, this representation is converted from the local format to a commonly accepted global format. Part 2 of this standard defines representations of a global format for Level 4-3 data exchanges. Part 4 of this standard defines representations of a global format for Level 3-3 data exchanges.
2. Any object identification and enumeration which needs to be exchanged and is not the same on all systems can be converted from a local namespace to a principle namespace (step 3 in the figure) and the principle namespace to the local namespace (step 7 in the figure). This part of the standard defines a minimal set of services and message rules which can be used to translate local to principle and principle to local identifiers.

Example 1 Assume two applications, ALPHA and BETA, the ALPHA application initiates a data exchange with the BETA application, and BETA responds back to ALPHA. The format conversions are:

1. ALPHA’s local format is converted to a global format for the request data.
2. Local identifiers in ALPHA are mapped into a principle (global) namespace.
3. The message is sent from ALPHA to BETA.
4. Upon receipt of the message, BETA must map the principle (global) identifiers into BETA’s namespace.
5. The global format must be converted to BETA’s local format for the request data
6. If there is any response required, then the process is reversed.

Example 2 IDs for identified elements may be status codes, tag names, or equipment identifiers.

Example 3 Data which is represented in one element namespace, such as codes 1,2,3,4 may have a different namespace representation in another application, such as codes Ok, Done, Error, Delay.

1. Messages are transported from one application to another, either within the same computer environment or across computers. Transport mechanisms are defined in other standards, such as TCP/IP and Ethernet standards. Part 6 of this standard defines a set of message services for the exchange of messages between applications.
2. When data exchange information is received, there are specific rules that define what resultant data is to be returned. The transaction rules are defined in Part 5 of this standard (step 10 in the figure).



Figure 1 – Steps in application-to-application communication

The part does not specify what technology an underlying system would use to implement the Alias services. Possible implementations include web services using a SOAP (Simple Object Access Protocol) interface, an HTTP/HTTPS implementation using a REST (Representational State Transfer) interface, or an API defined interface. This part defines the basic services with the assumption that specific technology implementations will be defined in companion specifications.

**ENTERPRISE-CONTROL SYSTEM INTEGRATION**

**Part 7: Alias service model**

# Scope

This part defines technology independent services and messages for associating and mapping aliases (equivalent identifiers) that are exchanged between manufacturing operations domain applications and applications in other domains. The identification of the organization and owner of each system and the lifecycle management of the equivalent object namespaces is outside the scope of this part.

# References

[1] ANSI/ISA-95.00.01-2010 (IEC 62264-1 Mod), Enterprise-Control System Integration – Part 1: Models and Terminology

[2] ANSI/ISA-95.00.02-2010 (IEC 62264-2 Mod), Enterprise-Control System Integration – Part 2: Object Models and Attributes

[3] ANSI/ISA-95.00.04-2012, Enterprise-Control System Integration – Part 4: Object Models and Attributes of Manufacturing Operation Management

[4] ANSI/ISA-95.00.05-2010 (IEC 62264-5 Mod), Enterprise-Control System Integration – Part 5: Business to Manufacturing Transactions

[5] ANSI/ISA-95.00.06, Enterprise-Control System Integration – Part 6: Message Service Model

# Definitions, abbreviations, and conventions

## Terms and definitions

3.1.1

alias

**equivalent identifier**

alternative ID for an object

3.1.2

alias directory

collection of equivalent identifications of objects

3.1.3

application

program or system that defines a namespace for IDs

NOTE: This defines the scope of where the ID namespace is defined. This may be a single instance of an application, multiple instances of the same application, applications using the same namespace from a single vendor, or a complete system that has applciations from different vendors that have agreed to a common namespace.

3.1.4

category

organizational element used to group principle items and identifiers defined in namespaces

3.1.5

identifier

sequence of characters that uniquely identifies an object within a specific context

3.1.6

local identifier

identifier of an object or enumeration for a specific namespace

3.1.7

namespace

set of unique named identifiers for a specific applicaton or system

3.1.8

namespace item

unique ID that identifies an object or enumeration within a category for a specific namespace

3.1.9

principle item

commonly agreed to identifier of an object or enumeration for exchange between different namespaces

## Abbreviations

ASM Alias Service Model

ID Identifier

## Conventions

Italics are used, beyond the use defined in ISO/IEC Directives Part2, to emphasize the 62264 specific meaning of terms. They are used for the following cases:

* Names of objects used in exchanged data
* Names of services

# Identifying objects and enumerations in data exchanges

## Converting data element IDs

Local IDs of data elements in an information exchange may not be globally known to all applications.

Example 1 Status values in one application may be defined as “**Good**” and “**Bad**”, but equivalent values may be defined as “**OK**” and “**NOK**” in another application or “**Pass**” and “**Fail**” in another application.

This Part defines a set of services that shall be used to provide an interface to a directory of equivalent IDs. It defines a common model for registration of identifiers and cross references of the identifiers for multiple systems. Different systems will often have different rules for the character sets available for use in identifiers and for the length of the identifiers.

Example 2 One system may allow A..Z, a..z, 0..9 in identifiers with a maximum length of 12, while another may also allow the use of the following characters “@/~\|\_-.” with a maximum length of 32 characters.

NOTE 1 This Part does not define how the directory should operate. It only defines a standard set of services to use the directory, and message exchanges for using the directory.

NOTE 2 This Part does not specify what technology an underlying system would use to implement the services.

NOTE 3 This Part does not specify any local representation of the directory, but defines an abstract model for the exchange of directory information using object models in the same format as Part 2 and 4, message transaction rules in the same format as Part 5 and a service model in the same format as Part 6.

## Principle item

It is important for system integration startup and for error analysis in message exchanges to be able to easily interpret the content of messages. For this reason the ASM has the concept of a principle item. Principle items should be defined as part of an engineering activity during an integration project. The services defined provide the support needed to perform the translations during message exchanges, and are not designed to provide maintenance access to a directory for creating of the directory or maintaining the directory.

NOTE 1 The assumption is an ASM service provider would provide the methods to create, maintain, and manage the directory.

NOTE 2 Often the principle item is the identifier from the *system of record*. The *system of record* is the authoritative data source for a given data element or piece of information.

Example The following tables list examples of principle to local identifier translations:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category ID | Principle ID | Alpha Namespace Item ID | Beta Namespace Item ID | Gamma Namespace Item ID |
| Material Status | **Pass** | Pass | réussir | pasar |
| Material Status | **Fail** | Fail | manquer | fallar |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category ID | Principle ID | Alpha Namespace Item ID | Beta Namespace Item ID | Gamma Namespace Item ID |
| Assemby Type | **Physical Assembly** | Physical Assembly | Physical | montaje físico |
| Assembly Type | **Logical Assembly** | Logical Assembly | Logical | montaje lógico |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category ID | Principle ID | Alpha Namespace Item ID | Beta Namespace Item ID | Gamma Namespace Item ID |
| Production Status | **Approved** | Available | Approved | aprobado |
| Production Status | **Released** | Ready | Released | liberado |
| Production Status | **Blocked** | Quarantined | Blocked | Obstruido |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category ID | Principle ID | Alpha Namespace Item ID | Beta Namespace Item ID | Gamma Namespace Item ID |
| Tag Names | **TIC-8106** | Unit1.TopTemp | TIC-8106 | TIC8106 |
| Tag Names | **LTC3\_AIT211** | AIT211 | Lone Tree PLC Analyzer | LTC3\_AIT211 |

## Identifier categories

Identifiers exist within a namespace of an application, and namespaces exit with categories in ASM. Each category defines a collection of unique principle items and a collection of namespaces. The categories will be normally defined by the applications and by the integration project. Many of the catrgories could be for property IDs. Other categories could be for statuses, and other enumerations not defined in the other parts of this standard. See Annex B for a list of recommended category names for exchanged enumerations.

## Alias directory use

ASM provides a method for applications to convert local identifiers to common identifiers (principle IDs), and convert common identifiers to local identifiers as part of a complete application-to-application communication protocol. This is illustrated as step 7 in Figure 2.



Figure 2 – Specifications in application-to-application communication

There are two models for translations, a set of standard services which translate lists of local to principle or principle to local identifiers, and a set of message transactions which can be used by an application to build a local cache of translated identifiers.

NOTE A local cache of translations can be used to reduce network communications and speed generation of exchange messages, but it does require that the ASM service provider supports the message exchange formats and that the application creating the cache also supports the message exchange formats.

# Alias directory object model

## Alias directory model

A collection of categories containing object and enumeration identifications shall be presented as an *alias directory*. An *alias directory* shall be made up of *categories*, with *categories* made up of *namespaces*, with *namespaces* made up of *namespace items*, with *namespace items* optionally containing *namespace item properties*. Alias directory *categories* also contain a set of *principle items*. Principle items may optionally contain *principle item properties*.

Figure 3 is the *alias directory* model.



Figure 3 – Alias directory model

## Alias directory elements

### Alias directory

An *alias directory* object is the container object for a set of categories.

EXAMPLE Within a company *alias directories* may be created for: testing, production use, local sites, global corporate information, areas, or departments.

Table 1 – Attributes of alias directory

| Attribute Name | Description | Examples |
| --- | --- | --- |
| ID | User supplied identifier of the *alias directory*.  An alias directory shall be unique within the scope of the ASM services. | “Test System”  “Qualified”  “Site Development” |
| Description | Description and expected use of the alias directory. | “Used for MES, LIMS, Historian interoperability testing”  “Used for qualified Level 3-Level 4 Integration”  “Used for site development” |

## Category

A *category* object is the container object for a set of *namespaces* and *principle items*. *Categories* define sets of related identifiers. See Annex B for a list of recommended *categories*. *Category* objects are made up of a collection of *principle items* and *namespace items*.

EXAMPLE Within a company:

* A *category* may be defined for Role Based Equipment Hierarchy Level Names (**Enterprise**, **Site**, **Area**, **Work Center**, **Work Unit** as the ISA-95 defined names), which may have different names (**Business**, **Site**, **Plant, Production Line**, **Production Unit**) in some applications.
* A *category* may be defined for equipment tag names.
* A *category* could be defined for material property names.
* A *category* could be defined for property value key IDs.

Table 2 – Attributes of category

| Attribute Name | Description | Examples |
| --- | --- | --- |
| ID | User supplied identifier of the *category*.  The ID shall be unique within the scope of the *alias directory*. | “Tagnames”  “Batch Status”  “Personnel Classes”  “Material Properties” |
| Description | Description and expected use of the *category*. | “SCADA, asset, historian, and network tag names”  “Equivalent batch status codes”  “Equivalent personnel class ID’s”  “Equivalent material property names” |
| CategoryOwner | Identification of the source or owner of the *category*. This may define the organization, specification, or part of a specification for which the *category* is defined. | “MIMOSA OSA-EAI V3”  “ISA 88.01”  “ISA-95.02.2010”  “IEC 62264-02 ed2”  ‘ChemCompany.RefineryModelV2.1’ |

## Principle item

A *principle item* is the container object for definition of a unique identifier within a category. A *principle item* may contain property definitions.

Table 3 – Attributes of principle item

| Attribute Name | Description | Examples |
| --- | --- | --- |
| ID | An identifier used as in an exchange message or to relate *namespace items* in different *namespaces*.  Each ID shall be unique within the scope of a *category*. | Good  Bad,  TIC-8106 |
| Description | Additional description of the *principle item*. | “Batch status is acceptable”  “Lot status is unacceptable” |

## Principle item property

A *principle item property* defines a property of a *principle item*. They are included in the situation in which a small set of properties may be further needed to distinguish *principle items*, and are not intended to be a global property master registry.

Table 4 – Attributes of principle item property

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Description** | **Examples** |
| ID | Identification of the *principle item property*.  Each ID shall be unique within the *principle item*. | ParentIdentifierID  ChildIdentifierID |
| Description | User defined description of the property | “The ID for a parent object of the identifier in the principle’s hierarchy”  “The ID for child objects of the identifier in the principle’s hierarchy” |
| Value | 1 or more string-serialized values of the property. | “FIC5565”  “Buiilding 5, Line 16”  [22,56,1,789] |

NOTE The format for values in value attributes is not defined in this part and will be defined by implementations of the standard.

The following syntax, defined in an EBNF notation from ISO 14977, may be used to represent single element values, range specifications, arrays of values, and a set of allowed values as delimited text strings.

<value> ::= <simpleValue>

| <arrayValue>

| <rangeValue>

| <seriesValue>

<arrayValue> ::= “[“ <arrayElement> \*( “,” <arrayElement> ) “]”

<rangeValue> ::= “{“ <rangeElement> \*( “,” <rangeElement> “}”

<seriesValue> ::= “<” <simpleValue> \*( “,” <simpleValue> ) “>”

<arrayElement> ::= <simpleValue> | <arrayValue>

<rangeElement> ::= <simpleValue> “..” <simpleValue>

<simpleValue> ::= string

## Namespace

A *namespace* object is the container object for a set of identifiers from a specific application that is involved in a data exchange.

Table 5 – Attributes of namespace

| Attribute Name | Description | Examples |
| --- | --- | --- |
| ID | Identification of the *namespace* or system for the identifier.  Each ID shall be unique within the scope of a *category*. | “AAA MES System  “ZZZ Operations Support System” |
| Description | Additional description of the *namespace* or system. | “AAA MES System Version 6.02” |
| NamespaceOwner | Organization which has responsibility for the *namespace* or system. | “Plant 5 Manufacturing IT Dept.”  “XXX Support Group” |

## Namespace item

A *namespace item* defines an entry within a *category* for a specific *namespace*.

EXAMPLE The tag **TC101** in system A may be the equivalent of tag **UNIT101.TOP\_TEMP** in system B.

Table 6 – Attributes of namespace item

| Attribute Name | Description |
| --- | --- |
| ID | User supplied ID of an item in the *namespace*.  Each ID shall be unique within the scope of a *namespace* within a *category*. |
| Description | Description of the *namespace item*. |
| PrincipleItemID | The ID of the identifier to be used for exchanges.  The *principle ID* associated with a PrincipleMap flag set to TRUE shall be unique within the scope of the category.  If empty, then a *namespace* ID has not yet been linked to a Principle ID. |
| Inactive | Boolean flag where FALSE indicates the identifier is active and available for use while TRUE indicates the identifier is inactive.  Example: Inactive entries may be:   1. Data that has been entered in the category system but is not yet available or not yet in use. 2. An identifier that is no longer valid in the category. |
| PrincipleMap | Boolean flag. Only one *namespace* ID within a *namespace* and for a specific Principle ID may have PrincipleMap set to TRUE. On a conversion from principle ID to local ID, if there are multiple choices for a namespace ID, then the ID with PrincipleMap set to TRUE is returned. |

Example Assume there is a unique mapping of a *namespace* ID to a *principle* ID, but there can be multiple *namespace* IDs associated with a single *principle* ID. The PrincipleMap is used to select which *namespace* ID to use when retrieving the *namespace* ID given a *principle* ID. Category: TANK\_FARM\_LEVEL\_STATUS

Principle IDs: (TANK\_EMPTY, TANK\_USED)

Namespace: APPLICATION-ALPHA,

SetPrincipleItemID: (ID=LEVEL\_LOW, PrincipleMap=TRUE, PrincipleItemID=TANK\_EMPTY)

SetPrincipleItemID (ID=LEVEL\_MEDIUM, PrincipleMap=FALSE, PrincipleItemID=TANK\_USED)

SetPrincipleItemID (ID=LEVEL\_HI, PrincipleMap=TRUE, PrincipleItemID=TANK\_USED)

GetPrincipleItemID (LEVEL\_LOW) 🡪 TANK\_EMPTY

GetPrincipleItemID (LEVEL\_MEDIUM) 🡪 TANK\_USED

GetPrincipleItemID (LEVEL\_HI) 🡪 TANK\_USED

GetLocalId (TANK\_EMPTY) 🡪 LEVEL\_LOW

GetLocalId (TANK\_USED) 🡪 LEVEL\_HI

Although two Local ID values have the same Principle ID assignments, Principle ID TANK\_USED mapped to Local ID LEVEL\_HI, because of the PrimaryMap=TRUE setting. (The LEVEL\_MEDIUM Local ID has a PrimaryMap=FALSE setting).

## Namespace item property

A *namespace item property* object defines a property of a *namespace item*. Properties may be used to help identify identifiers or to discover a principle item. The properties should be a small set of properties that may be needed to link systems together, and are not intended to be a global property master registry.

Table 7 – Attributes of namespace item property

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Description** | **Examples** |
| ID | User defined identification of the *namespace item property*.  The ID shall be unique within the *namespace item*. | ParentIdentifierID  ChildIdentifierID |
| Description | User defined description of the property | “The ID for the parent object of the identifier in the namespaces’s hierarchy”  “The ID for child objects of the identifier in the namespaces’s hierarchy” |
| Value | 1 or more string-serialized values of the property. | “FIC5565”  “Buiilding 5, Line 16” |

# Alias services

## Alias service elements

The ASM model defines a set of standard services that can be used to:

1. Obtain the *principle items* from a set of local *namespace items* for an application.
2. Obtain the local *namespace items* for an application from a set of *principle items*.
3. Obtain *namespace items* for a target application, given a source application and the source’s *namespace items*
4. Query the *alias directory* for information about *categories*, *namespaces*, and *namespace items*.
5. Create a set of *namespace items* for a given *namespace*
6. Create a set of *principle items* for a *category*
7. Create equivalent *namespace items* from two *namespaces*.

## Security

### Secure access

Security in alias services shall be defined as authenticated access to ASM services.

NOTE Security using the message service model is defined in Part 7 of this standard.

### Security Tokens

Alias service security shall be managed through security tokens.

Security tokens are assigned to alias directories.

Security tokens on *alias directories* may be optionally added by the users of the ASM services.

NOTE While the ASM service provider is required to provide security, the final users of the ASM services may decide not to assign any security tokens to one or more *alias directories*, in which case the *alias directory* may be accessed without any authentication control.

Security tokens shall be used by applications when accessing services on an *alias directory*. If the application provided security token does not match a security token assigned to the *alias directory*, then no information is returned.

Security tokens shall be exchanged in an out-of-band communication channel, such as manual exchange of tokens, or electronic exchange through a secure point-to-point channel.



Figure 4 – Security of identifier registries

### Security token format

The security token format shall be defined in the ASM implementation specification. Different implementations may have different methods and formats for the security tokens.

### ASM service provider implementations

1. All ASM service providers shall implement security tokens.
2. ASM service providers may limit the ability to use the ASM services to approved applications, servers, or domains in order to increase security.
3. While there is a requirement that the services provide security services, there is no requirement that a specific implementation use the services.

Example 1 A system may share information across companies through open Internet channels. In this case an ASM *service provider* implementation should provide a strong security token system through a public key mechanism with specific security token assigned to specific communicating companies.

Example 2 A system may be entirely contained within a secure environment behind both corporate and operations firewalls. In this case the user may decide to not assign security tokens to channels and rely on other measures to ensure security.

## ASM service type definitions

Table 8 contains the type definitions that are associated with the service definitions.

Table 8 – ASM type definitions

|  |  |
| --- | --- |
| Type | Definition |
| AliasDirectory | Contains an *alias directory* object. May optionally contain *category* definitions, which may optionally contain *namespace* definitions, which may optionally contain *namespace item* definitions, which may optionally contain *namespace item property* definition and property values. |

## ASM service returns and faults

Table 9 contains the service returns and faults for the ASM service definitions.

Faults should contain a human readable explanation.

Table 9 – ASM service returns and fault definitions

|  |  |
| --- | --- |
| Type | Definition |
| Directory Fault | Error returned when an *alias airectory* ID is invalid or the application does not have the appropriate security token to access the directory. |
| Category Fault | Error returned when a *category* ID is invalid. |
| Source Fault | Error returned when a source *namespace* ID is invalid. |
| Identifier Fault | Error returned when a *namespace identifier* ID is invalid. |
| Target Fault | Error returned when a target *namespace* ID is invalid. |

## ASM mapping services

### Get namespace IDs

Table 10 defines the *get namespace IDs* service. This service is used to convert a set of *principle item* IDs into a *namespace item* ID.

NOTE This service may be used to create a local cache of information, but care must be taken to refresh the cache at regular intervals or on an out-of-band event that indicates new data is available.

Table 10 – Get namespace IDs

|  |  |
| --- | --- |
| **Name** | Get namespace IDs |
| **Function** | Get a set of *namespace item* IDs for a specified *alias directory*, *category*, and *namespace* given a list of *principle item* IDs.  If the *alias directory* does not exist then a *Directory Fault* is returned.  If the specified *alias directory* is assigned security tokens and none of the specified Security Tokens match a token assigned to the specified *alias directory*, then a *Directory Fault* is returned.  If the *category* does not exist then a *Category Fault* is returned.  If the *namespace* does not exist then a *Source Fault* is returned.  If a *namespace item* ID for a *principle item* ID does not exist, then the associated *namespace item* ID is returned as an empty string. |
| **Input Parameters** | * Alias Directory ID * Category ID * Namespace ID * List of principle item IDs * Optional list of Security Tokens |
| **Returns** | * List of namespace item IDs * Success or error criteria |

### Get principle IDs

Table 11 defines the *get principle ID* service. This service is used to convert a set of *namespace item* IDs into the *principle item* ID *namespace*.

NOTE This service may be used to create a local cache of information, but care must be taken to refresh the cache at regular intervals or on an out-of-band event.

Table 11 – Get principle IDs

|  |  |
| --- | --- |
| **Name** | Get principle IDs |
| **Function** | Get a set of *principle item* IDs for a specified *alias directory*, *category*, and *namespace* given a list of *namespace item* IDs.  If the *alias directory* does not exist then a *Directory Fault* is returned.  If the specified *alias directory* is assigned security tokens and none of the specified Security Tokens match a token assigned to the specified *alias directory*, then a *Directory Fault* is returned.  If the *category* does not exist then a *Category Fault* is returned.  If the *namespace* does not exist then a *Source Fault* is returned.  If a *namespace item* ID is null, then a *Source ID* Fault is returned.  If a *namespace item* ID does not have a *principle item ID* mapped, then the associated *principle item ID* returned is an empty string. |
| **Input Parameters** | * Alias Directory ID * Category ID * Namespace ID * List of identifiers without PrincipleItemIDs * Optional list of Security Tokens |
| **Returns** | * List of identifiers with PrincipleItemIDs * Success or error criteria |

### Get equivalent IDs

Table 12 defines the *get equivalent IDs* service. This service is used to convert a set of *namespace item* IDs from a source *namespace* into the *namespace item* IDs for a target *namespace*.

NOTE This service may be used to create a local cache of information, but care must be taken to refresh the cache at regular intervals or on an out-of-band event.

Table 12 – Get equivalent IDs

|  |  |
| --- | --- |
| **Name** | Get equivalent IDs |
| **Function** | Get a set of *namespace item* ID values for a target *namespace*, for a specified *alias directory*, *category*, source *namespace* and list of *namespace item* IDs. This service performs the equivalent of a Get Principle ID call using the source *namespace*, followed by a Get Namespace ID call using the target *namespace*.  If the *alias directory* does not exist then a *Directory Fault* is returned.  If the specified *alias directory* is assigned security tokens and none of the specified Security Tokens match a token assigned to the specified *alias directory*, then a *Directory Fault* is returned.  If the *category* does not exist then a *Category Fault* is returned.  If the source *namespace* does not exist then a *Source Fault* is returned.  If the target *namespace* does not exist then a *Target application Fault* is returned.  If a *namespace item* ID is null, then a *Source ID* Fault is returned.  If a *namespace item* ID does not not have an equivalent ID in the target *namespace*, then the ID returned is an empty string. |
| **Input Parameters** | * Alias Directory ID * Category ID * Source namespace ID * Target namespace ID * List of identifiers of the source namespace * Optional list of Security Tokens |
| **Returns** | * List of identifiers from the target namespace * Success or error criteria |

### Get alias directory information

Table 13 defines the *get* *alias directory* information service. This service shall return information about the *alias directory* in an ASM service provider independent format.

NOTE 1 The get *alias directory* may be used to return information about elements of an *alias directory*. The rules defined in Table 14 define what parts of the *alias directory* are returned based on the information sent with the service.

NOTE 2 This service may be used to create a local cache of information, but care must be taken to refresh the cache at regular intervals or on an out-of-band event.

Table 13 – Get alias directory information

|  |  |
| --- | --- |
| **Name** | Get alias directory information |
| **Function** | Returns *alias directory* information from an ASM alias directory.  If the *alias directory* does not exist then a *Directory Fault* is returned.  If the specified *alias directory* is assigned security tokens and none of the specified Security Tokens match a token assigned to the specified *alias directory*, then a *Directory Fault* is returned. |
| **Input Parameters** | * Alias directory object * Optional list of Security Tokens |
| **Returns** | * Alias directory object * Success or error criteria |

Table 14 defines the rules that specify what information is returned from the get alias directory information service.

Table 14 – Get alias directory information rules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Alias Directory ID | Category ID | Namespace ID | Namespace Item ID | Namespace Item Property ID | Rule |
| Not defined | Not defined | Not defined | Not defined | Not defined | Returns all *alias directory* objects and all *category* objects in each alias directory. |
| Defined | Not defined | Not defined | Not defined | Not defined | Returns the identified *alias directory* object, all *category* objects in the *alias directory*, and all *namespace* objects in each *category*. |
| Defined | Defined | Not defined | Not defined | Not defined | Returns the identified *alias directory* object, the identified *category* objects in the *alias directory*, all *principle items* in the *category*, all *principle item properties* and property values of the principle objects, all *namespaces* in each *alias directory*, and all *namespace items* in each *namespace*. |
| Defined | Defined | Defined | Not defined | Not defined | Returns the identified *alias directory* object, the identified *category* objects in the *alias directory*, all *principle items* in the *category*, all *principle item properties* and property values of the principle items, the identified *namespace*, all *namespace items* in the identified *namespace,* and all property and property values in all identifiers. |
| Defined | Defined | Defined | Defined | Not defined | Returns the identified *alias directory* object, the identified *category* object in the *alias directory*, all *principle items* in the *category*, the identified *namespace item* in the *category*, the identified *namespace items*, and all property and property values in the identified *namespace items*. |
| Defined | Defined | Defined | Defined | Defined | Returns the identified *alias directory* object, the identified *category* object in the directory, the identified *namespace* in the *category*, the identified *namespace*, the identified *namespce item* in the *namespace*, the identified properties for the identified *namespace item properties*. |

### Set namespace items

Table 15 defines the *set namespace items* service. This service is used to define a set of *namespace items* for a given *namespace* with optional *principle item* IDs and PrincipleMap flag.

Table 15 – Set namespace items

|  |  |
| --- | --- |
| **Name** | Set namespace items |
| **Function** | Add *namespace item* IDs for a specified *alias directory*, *category*, and *namespace*.  The *namespace items* may define an optional *principle item* ID.  The *namespace items* may define *namespace item properties*.  If the *alias directory* does not exist then a *Directory Fault* is returned.  If the specified *alias directory* is assigned security tokens and none of the specified Security Tokens match a token assigned to the specified *alias directory*, then a *Directory Fault* is returned.  If the *category* does not exist then the identified *category* is added.  If the *namespace* does not exist then the identified *namespace* is added.  If the *namespace item* ID already exists, then no error is returned.  If the *namespace item* ID already exists and a *namespace item property* does not exist, then the *namespace item property* is added.  If the *namespace item* ID already exists and a *namespace item property* already exists, then the *namespace item property* value is replaced. |
| **Input Parameters** | * Alias Directory ID * Category ID * Namespace ID * List of namespace item objects * Optional list of Security Tokens |
| **Returns** | * Success or error criteria |

### Set principle items

Table 16 defines the *set principle items* service. This service is used to define a set of *principle items* for a given *category*.

Table 16 – Set principle items

|  |  |
| --- | --- |
| **Name** | Set principle items |
| **Function** | Add *principle items* for a specified *alias directory* and *category*.  The *principle items* may define *principle item properties*.  If the *alias directory* does not exist then a *Directory Fault* is returned.  If the specified *alias directory* is assigned security tokens and none of the specified Security Tokens match a token assigned to the specified *alias directory*, then a *Directory Fault* is returned.  If the *category* does not exist then the identified *category* is added.  If the *principle item* already exists, then no error is returned.  If the *principle item* already exists and a *principle item property* does not exist, then the *principle item property* is added.  If the *principle item* already exists and a *principle item property* already exists, then the *principle item property* value is replaced. |
| **Input Parameters** | * Alias Directory ID * Category ID * List of principle items * Optional list of Security Tokens |
| **Returns** | * Success or error criteria |

### Set equivalent identifiers

Table 17 defines the *set equivalent identifiers* service to set equivalent *namespace items* from two *namespaces*. This service is used to define a set of *namespace items* in a source *namespace* and in a target *namespace*, and to create *principle items* that link the two together.

Table 17 – Set equivalent identifiers

|  |  |
| --- | --- |
| **Name** | Set equivalent identifiers |
| **Function** | Add *namespace items* to the source *namespace*, the target *namespace*, and optionally create *principle items* to link the identifiers.  If the *alias directory* does not exist then a *Directory Fault* is returned.  If the specified *alias directory* is assigned security tokens and none of the specified Security Tokens match a token assigned to the specified *alias directory*, then a *Directory Fault* is returned.  If the *category* does not exist, then the identified *category* is added.  If the source *namespace* does not exist, then the identified *namespace* is added.  If the target *namespace* does not exist, then the identified *namespace* is added.  If a *namespace item* has a PrincipleItemID defined, then the principle ID is used as the created *principle item*, else if a target *namespace item* has a PrincipleItemID defined, then the PrincipleItemID is used as the created *principle item*, else the ASM implementation creates a principle ID. |
| **Input Parameters** | * Alias Directory ID * Category ID * Source namespace ID * List of source namespace items * Target namespace ID * List of target namespace items * Optional list of Security Tokens |
| **Returns** | * Success or error criteria |

NOTE The algorithm used by the ASM implementation to create a principle ID is not defined.

# Alias directory model transactions

## Alias directory elements

The message definitions assume that directory information may be accessed from one starting point; an alias directory, as identified by the dotted collection in Figure 5.

Message exchanges should follow the channel and topic model defined in Part 6 of this standard. See Annex B for a list of recommended category names.



Figure 5 – Object grouping for the alias directory model

## Alias directory verbs

Only the GET and SYNC verbs defined in Part 5 shall be valid for use with an *alias directory* noun.

## Alias directory verb actions

The actions performed on an *alias directory* object are defined in Table 18.

Table 18 – Alias directory verb actions

| **Verb** | **Alias directory Action** |
| --- | --- |
| **GET** | Shall define a request that the receiver is to return, in a SHOW message, information about the *alias directory*, as specified by information sent in GET message, as specified in the Get alias directory service in Table 14. |
| **PROCESS** | not defined |
| **CHANGE** | not defined |
| **CANCEL** | not defined |
| **SYNC ADD** | Shall define a request that the receiver should add the specified identifiers to their local cache of identifers. The category and namespace are identified in the SYNC ADD message. |
| **SYNC CHANGE** | Shall define a request that the receiver should change the identifiers in their local cache. The category and namespace are identified in the SYNC CHANGE message. |
| **SYNC DELETE** | Shall define a request that the specified identifers are no longer valid. The category and namespace are identified in the SYNC DELETE message. |

NOTE: When maintaining a local cache it will usually only be for one specified namespace, so receivers of the message may decide to filter out messages based on the namespace specification.

# Completeness, compliance and conformance

## Completeness

The number of services and messages supported, as defined in Table 19 shall determine the degree of completeness of a specification or application.

## Compliance

Any assessment of the degree of compliance of a specification shall be qualified by the following:

1. the use of the terminology defined in this part;
2. the use of the services defined in this part;
3. the use of the protocol for each supported transaction;
4. a statement of the degree to which they then conform partially or totally to definitions and transaction names.

In the event of partial compliance, areas of non-compliance shall be explicitly identified.

## Conformance

Any assessment of the degree of conformance of an application shall be qualified by the following:

1. documentation of the services and transactions, as listed in Table 19,
2. documentation of the transaction rules conformed to,
3. documentation of services supported,

In the event of partial conformance, areas of non-conformance shall be explicitly identified.

Suppliers of applications shall use Table 19 or an equivalent to document their supported transactions.

Suppliers of applications shall document if the application can perform the role of the information user, if the application can perform the role of the information provider, if the application can perform the role of information sender, and if the application can perform the role of information receiver.

Suppliers shall document their support for wildcards in the appropriate transactions.

Table 19 – Supported services and verb/noun transactions

|  |  |
| --- | --- |
| Service | Get namespace IDs |
| Service | Get PrincipleItemIDs |
| Service | Get alias directory information |
| Service | Set namespace items |
| Service | Set principles identifiers |
| Message | Sync alias directory |
| Message | Get alias directory |

1. – (Normative) Recommended value properties

There are recommend property names that should be used to identify commonly understood relationships between entities.

## ParentIdentifierID

The ParentIdentifierID contains the set of IDs for parent objects of the identifier in the source’s namespace hierarchy. Multiple parent objects are specified by multiple PropertyValues.

## ChildIdentifierID

The ChildIdentifierID contains the set of IDs for child objects of the identifier in the source’s namespace hierarchy. Multiple child objects are specified by multiple PropertyValues.

## PossibleEquivalentIdentifierID

The PossibleEquivalentIdentifierID contains a set of target entities which are possibly equivalent to the identifier. This allows for automated equivalency determination. Each returned target identifier contains the following set of information:

1. ID
2. Namespace ID
3. PercentLikelihood [Optional]

## EffectiveStartDate

The EffectiveStartDate contains the date and time at which the namespace item ID or principle item ID becomes effective and the item be used for mapping.

## EffectiveEndDate

The EffectiveEndDate contains the date and time at which the namespace item ID or principle item ID becomes ineffective and the item maynot be used for mapping.

1. – (Normative) Recommended categories

Identifiers that refer to enumerated elements defined in the ISA-95 object models should use the following category names.

Table 20 – Standard category names

| Category name | Object model(s) | Attribute |
| --- | --- | --- |
| Assembly Relationship | Material class  Material definition  Material lot  Material sublot  Operations material bill item  Material specification  Material requirement  Material actual  Material capability  Material segment specification | Assembly Relationship  Assembly Relationship  Assembly Relationship  Assembly Relationship  Assembly Relationship  Assembly Relationship  Assembly Relationship  Assembly Relationship  Assembly Relationship  Assembly Relationship |
| AssemblyType | Material class  Material definition  Material lot  Material sublot  Operations material bill item  Material specification  Material requirement  Material actual  Material capability  Material segment specification | Assembly Type  Assembly Type  Assembly Type  Assembly Type  Assembly Type  Assembly Type  Assembly Type  Assembly Type  Assembly Type  Assembly Type |
| Capacity Type | Operations capability  Personnel capability  Equipment capability  Material capability  Physical asset capability  Process segment capability  Work capability  Work master capability | Capacity Type  Capacity Type  Capacity Type  Capacity Type  Capacity Type  Capacity Type  Capacity Type  Capacity Type |
| Connection Type | Resource network connection type | Type |
| Confidance Factor | Operations capability  Personnel capability  Equipment capability  Material capability  Physical asset capability  Work capability | Confidence Factor  Confidence Factor  Confidence Factor  Confidence Factor  Confidence Factor  Confidence Factor |
| Dependency Type | Process segment dependency  Operations segment dependency | Dependency Type  Dependency Type |
| Dispatch Status | Job order | Dispatch Status |
| Equipment Level | Equipment class  Equipment | Equipment Level  Equipment Level |
| Equipment Test Result | Equipment capability test result | Result |

Table 20 - Continued

| Category Name | Object model(s) | Attribute |
| --- | --- | --- |
| Equipment Use | Equipment segment  Equipment specification  Equipment requirement  Equipment actual  Equipment capability | Equipment Use  Equipment Use  Equipment Use  Equipment Use  Equipment Use |
| Job Command | Job order | Command |
| Material Lot Status | Material Lot  Material Sublot | Status  Status |
| Material Test Result | Material test result | Result |
| Material Use | Material segment  Material specification  Operations material bill item  Material requirement  Material actual  Material capability | Material Use  Material Use  Use Type  Material Use  Material Use  Material Use |
| Operations Type | Process segment | Operations Type |
| Operations Type | Operations definition  Operations segment  Operations schedule  Operations request  Segment requirements  Operations performance  Operations request  Segment response | Operations Type  Operations Type  Operations Type  Operations Type  Operations Type  Operations Type  Operations Type  Operations Type |
| Performance State | Operations performance  Operations request | Performance State  Response State |
| Personnel Use | Personnel segment  Personnel specification  Personnel requirement  Personnel actual  Personnel capability | Personnel Use  Personnel Use  Personnel Use  Personnel Use  Personnel Use |
| Physical Asset Test Result | Physical Asset capability test result | Result |
| Physical Asset Use | Physical asset segment  Physical asset specification  Physical asset requirement  Physical asset actual  Physical asset capability | Physical Asset Use  Physical Asset Use  Physical Asset Use  Physical Asset Use  Physical Asset Use |
| Priority | Job order  Work alert definition  Work alert | Priority  Priority  Priority |
| Qualification Test Result | Qualification test result | Result |
| Relationship Form | Resource relationship network | Relationship Form |
| Relationship Type | Resource relationship network | Relationship Type |
| Resource Type | From resource reference  To resource reference | Resource Type  Resource Type |
| Scheduled State | Operations schedule  Operations request | Scheduled State  Request State |

Table 20 - Continued

| Category Name | Object model(s) | Attribute |
| --- | --- | --- |
| Work Alert Category | Work alert | Category |
| Work Type | Work definition  Work schedule  Work request  Job list  Job order  Work performance  Work response  Job response | Work Type  Work Type  Work Type  Work Type  Work Type  Work Type  Work Type  Work Type |

The activities defined in Part 3 of this standard could also be used to define *category* names, such that services to support the activities would have equivalent identifiers in *categories* defined by the primary source or sink activity (e.g. properties used in Production Scheduling, Inventory Analysis, Quality Resource Management, …).

Identifiers that refer to object classes and properties defined in the ISA-95 object models should use the following category names.

Table 21 - Object class and property categories

| Category name | Object / element name |
| --- | --- |
| Personnel Class | Personnel class / ID  Personnel segment specification / Personnel class  Personnel specification / Personnel class  Personnel requirement /Personnel class  Personnel actual / Personnel class  Personnel capability /Personnel class |
| Personnel Property | Personnel class property / ID  Person property / ID  Personnel segment specification property / ID  Personnel specification property / ID  Personnel requirement property / ID  Personnel actual property / ID  Personnel capability property / ID |
| Qualification Test | Qualification test / ID |
| Equipment Class | Equipment class / ID  Equipment segment specification / Equipment class  Equipment specification / Equipment class  Equipment requirement / Equipment class  Equipment actual / Equipment class  Equipment capability / Equipment class |

Table 21- Continued

| Category Name | Object / Element Name |
| --- | --- |
| Equipment Property | Equipment class property / ID  Equipment property / ID  Equipment segment specification property / ID  Equipment specification property / ID  Equipment requirement property / ID  Equipment actual property / ID  Equipment capability property / ID |
| Equipment Capability Test Specification | Equipment capability test specification / ID |
| Physical Asset Class | Physical asset class / ID  Physical asset segment specification / Physical asset class  Physical asset specification / Physical asset class  Physical asset requirement / Physical asset class  Physical asset actual / Physical asset class  Physical asset capability / Physical asset class |
| Physical Asset Property | Physical asset property / ID  Physical asset segment specification property / ID  Physical asset specification property / ID  Physical asset requirement property / ID  Physical asset actual property / ID  Physical asset capability property / ID |
| Physical Aasset Capability Test Specification | Physical asset capability test specification / ID |
| Material Class | Material class / ID  Material segment specification / Material class  Material specification / Material class  Material requirement / Material class  Material actual / Material class  Material capability / Material class  Operations material bill item / Material class |
| Material Definition | Material definition / ID  Material segment specification / Material definition  Material specification / Material definition  Material requirement / Material definition  Material actual / Material definition  Material capability / Material definition  Operations material bill item / Material definition |
| Material Property | Material class property / ID  Material definition property / ID  Material lot property / ID  Material segment specification property / ID  Material specification property / ID  Material requirement property / ID  Material actual property / ID  Material capability property / ID |
| Material Test Specification | Material test specification / ID |

Table 21- Continued

| Category name | Object / element name |
| --- | --- |
| Process Segment | Process segment / ID  Operations segment / Process segment  Process segment capability / Process segment |
| Process Segment Parameter | Process segment parameter / ID |
| Operations Definition | Operations definition / ID  Operations request / Operations definition ID  Segment requirement / Operations definition ID  Operations response / Operations definition ID  Segment response / Operations definition ID |
| Operations Segment | Operations segment / ID |
| Work Definition | Work definition / ID  Operations segment / Work definition ID |
| Segment Parameter | Segment specification / ID  Segment parameter / ID |
| Segment Data | Segment data ID |
| Work Definition | Work definition ID  Job order attributes / Work master ID  Work capability / Work master |

1. – Use scenarios
2. Figure 6 illustrates the use of the services in message exchanges. Application “A” would retrieve a set of *principle item* IDs from a set of *namespace item* IDs. The *principle item* IDs are used in Application “A” to create a message that is sent to application “B”. Application “B” then gets its *namespace item* IDs from the *principle item* IDs, and processes the message using its *namespace item* IDs.



Figure 6 – ASM simple example using services

1. Figure 7 illustrates the use of messages to obtain aliases. Application “A” sends a “GET Alias Directory” and receives a “SHOW Alias Directory” with the *principle items*. Application “A” creates a message with the *principle items* and sends it to application “B”. Application “B: receives the message, then application “B” sends a “GET Alias Directory” and receives a “SHOW Alias Directory” with local *namespace item* IDs. Application “B” then converts the *principle items* in the message to the local *namespace items* and processes the message.



Figure 7 – ASM simple example using messages

1. Figure 8 illustrates the use of services to create local caches of *principle item* and local *namespace item* ID equivalances. Application “A” uses the Get Principle IDs service to obtain a set of mappings of *principle item* IDs to A’s local *namespace item* IDs. The mapping is then maintained in a cache in Application A”. Application “B” uses the the Get Principle IDs service to obtain a set of mappings of *principle item* IDs to B’s local *namespace item* IDs, and also creates a local cache. When Application “A” sends a message it converts its local *namespace item* IDs to *principle item* IDs. When Application “B” receives a message it converts the *principle item* IDs to local *namespace item* IDs and processes the mesage.



Figure 8 – ASM cache example using services

1. Figure 9 illustrates the use of services to create local caches of *principle item* and local *namespace item* ID equivalances. Application “A” uses the Get Principle IDs service to obtain a set of mappings of *principle item* IDs to A’s local *namespace item* IDs. The mapping is then maintained in a cache in Application A”. Application “B” uses the the Get Principle IDs service to obtain a set of mappings of *principle item* IDs to B’s local *namespace item* IDs, and also creates a local cache. When Application “A” sends a message it converts its local *namespace item* IDs to *principle item* IDs. When Application “B” receives a message it converts the *principle item* IDs to local *namespace item* IDs



Figure 9 – ASM cache example using messages

1. Figure 10 illustrates applications of the ASM using services and messages, with Application “A” using messages to create a local cache and Application “B” using services to obtain local IDs from principle IDs on an as-needed basis.



Figure 10 – ASM multiple methods

1. Using the PrincipleItemID*set principle items* and *set namespace items* with GUIDs for the Principle item IDs

NOTE 1 A GUID is a Globally Unique Identifier that is a unique reference number used as an identifier. The term GUID typically refers to various implementations of the universally unique identifier (UUID) defined in the ISO/IEC 9834-8:2005 standard. GUIDs are usually stored as 128-bit values, and are commonly displayed as 32 hexadecimal digits with groups separated by hyphens, such as {21EC2020-3AEA-4069-A2DD-08002B30309D}.

* + Application ALPHA is configured to use a specific *alias directory* ID and *category* ID, for example *alias directory* ID = “Production”, *category* ID = “MIMOSA CCOM Asset”.
  + ALPHA creates the new local equipment record.
  + ALPHA also generates a MIMOSA CCOM GUID.
  + ALPHA calls *Set Principle items* with the *alias directory* ID, the *category* ID, and the GUID.
  + ALPHA calls *Set Namespace Items* with the *alias directory* ID, the *category* ID, and the local identifier also set to GUID

NOTE 2 In this example ALPHA’s LocalID matches the PrincipleItemID.

* + ALPHA publishes the creation of the equipment using the MSM model (Part 6) and the *SyncEquipmentDefinition* message on a “*MIMOSA CCOM Asset*” channel.
  + Application BETA had been previously configured to use a specific *alias directory* ID and *category* ID, for example *alias directory* ID = “Production”, *category* ID = “MIMOSA CCOM Asset”.
  + BETA had previously subscribed to the “MIMOSA CCOM Asset” channel.
  + BETA receives the *SyncEquipmentDefinition* message.
  + BETA queries the ASM using the *get namespace IDs* service and the GUID.
  + The *get namespace IDs* service returns with an empty string for the LocalID.
  + Because an empty string is returned, BETA understands that this is a new object. BETA creates a new local equipment record.
  + BETA calls the s*et namespace IDs* service using the GUID as the PrincipleItemID, and its newly created local ID as the LocalID.

1. One system with a single name “product” and another with a rich structure of product types.

A MOM (Manufacturing Operations Management) system has a scheduling system, a LIMS (Laboratory Information Management System) application and a WMS inventory management system.

* The LIMS system processes information about quality and chemistry for multiple sample systems, and material (liquid) that moves through one of these sample systems is identified by the name “product”.
* The MOM scheduling system and the WMS inventory management system capture information about the multiple products which are analyzed through the LIMS application and temporarily processed in storage tanks, including the product name, mass or volume, tank status and other attributes.
* The MOM scheduling system uses a structure of product names, which includes the multiple grades of product, using the multiple names of “high grade product”, “medium grade product”, and “low grade product”.
* The inventory management system uses the same multiple names, but without a structure for these names (it uses other data structures for each tank).
* During the manufacturing operations used for high-grade product, the MOM system must associate the names “*high grade product*” and “*product*”, and it must associate the scheduling system structure (product with multiple product grades) with the LIMS data structure and the inventory management system structure.
* Only the MOM system as a whole “knows” that a high-grade product is being manufactured at the moment. This knowledge is used to select the correct association within the alias directory. The association is at the attribute name level, not at the structure level.

1. In a federation of systems, each system has its own database and unique rules for identifying objects and object structures. Each system uses a different name for the same object structure element (ID, Tagname, Identifier, …). The ASM is used to convert each system’s name for the structure element into a PrincipleItemID (TAGNAME), and all exchanged messages use the structure element name of TAGNAME to identify the structure element.

1. – FAQ

This section defines frequently asked questions about this part of ISA-95.

**QUESTION:**

What is the need in view of existing standards like ISO 11179/19763 and ISO 15000?

**ANSWER:**

ISO/IEC 11179 helps guide the naming process within a set of standard rules for doing so. i.e. “Here is what you should name your data elements within your system.” That may have been accomplished by multiple parties on disparate systems that may exist within a company. We believe we see a gap here that Part 7 can help solve. What happens when these disparate systems come together? They may or may not have applied 11179 standard practices for naming identifiers within each of those systems. However, they will not be cleanly mapped to each other regardless. This part contributes a set of services that companies may implement, which allow a “decoder ring” that brings multiple disparate systems together at the L3-L4 messaging layer.

**QUESTION:**

Does this require a full new system to implement the services?

**ANSWER:**

The services are defined in such a way that it could be layered on to of ESBs (Enterprise Service Busses) that provide equivalent services, it could be layered on top of an LDAP (Lightweight Directory Access Protocol), it could be layered on top of a SQL database, or many other possible architectures.

In addition the services may be implemented using SOAP (Simple Object Access Protocol) and web services, or through a REST (Representational State Transfer) implementation,