Integrating the Healthcare Enterprise

IHE IT Infrastructure
Technical Framework Supplement

XDS Metadata Update

Trial Implementation

Date: August 31, 2015

Author: ITI Technical Committee

Email: iti@ihe.net

Please verify you have the most recent version of this document. See here for Trial Implementation and Final Text versions and here for Public Comment versions.

Foreword

This is a supplement to the IHE IT Infrastructure Technical Framework 11.0. Each supplement
30   undergoes a process of public comment and trial implementation before being incorporated into
the volumes of the Technical Frameworks.

This supplement is published on August 31, 2015 for trial implementation and may be available
for testing at subsequent IHE Connectathons. The supplement may be amended based on the
results of testing. Following successful testing it will be incorporated into the IT Infrastructure
35   Technical Framework. Comments are invited and may be submitted at
http://ihe.net/ITI_Public_Comments.

This supplement describes changes to the existing technical framework documents.

"Boxed" instructions like the sample below indicate to the Volume Editor how to integrate the
relevant section(s) into the relevant Technical Framework volume.

40   *Amend Section X.X by the following:*

Where the amendment adds text, make the added text **bold underline**. Where the amendment
removes text, make the removed text **bold strikethrough**. When entire new sections are added,
introduce with editor's instructions to "add new text" or similar, which for readability are not
bolded or underlined.

45

General information about IHE can be found at: http://www.ihe.net.

Information about the IHE IT Infrastructure domain can be found at:
http://www.ihe.net/IHE_Domains.

Information about the structure of IHE Technical Frameworks and Supplements can be found at:
50   http://www.ihe.net/IHE_Process and http://www.ihe.net/Profiles.

The current version of the IHE Technical Framework can be found at:
http://www.ihe.net/Technical_Frameworks.

55

CONTENTS

4

Introduction

This supplement updates the XDS and XDR profiles to add support for the updating and deleting
of metadata.

One new actor and two new transactions are introduced. The Document Administrator is the
source of the new transactions. The Update Document Set transaction carries metadata updates
and the Delete Document Set transaction enables metadata deletion. These new capabilities are
assigned to a new actor and new transactions to enable tighter authentication/authorization
control over their use.

In XDS, Update Document Set and Delete Document Set are sent to the Document Registry and
updates are visible through the Registry Stored Query transaction. In XDR, Update Document
Set and Delete Document Set are sent to the Document Recipient and no query is possible due to
the nature of XDR.

Updates use the SubmissionSet object to anchor each submission following the same rules as the
Register Document Set transaction. But with update, each metadata object that accompanies the
SubmissionSet triggers an update to the registry contents. Updates take the following forms:

- New version of the metadata for a DocumentEntry or Folder object

- Changes in the availabilityStatus attribute of DocumentEntry, Folder, and Association
  objects

- Addition of Association objects

  Note: This is the first time the availabilityStatus attribute has been used on the Association object.

New attributes logicalID and Version are introduced from ebRIM 3.0 to manage the details of
versioning. A new attribute, documentAvailability, is introduced on the DocumentEntry object to
allow separate management of the status of a document in a repository. It is now possible to label
a document as being Offline. It still exists but cannot be retrieved at the current time.

The Patient ID attribute on DocumentEntry and Folder objects can be updated. This will enable
new approaches in the handling of Patient ID Merge/Link.

A key concept for versioning from ebRIM 3.0 is the logicalID or lid (ebRIM name) attribute.
The existing id attribute (entryUUID name is used in XDS) is unique for every metadata object
in the registry. The new logicalID attribute is also assigned to every metadata object but all
objects that are versions of the same logical object share the same value of logicalID. Different
versions of a logical object are differentiated by the version attribute that carries the version
number: 1, 2, 3 and so on.

The second key concept for versioning is that Associations point to a specific version of an
object. DocumentEntry objects are linked to a Folder object through HasMember Associations
showing they are part of the Folder. With the introduction of versioning, this description is
updated to show that a version of a DocumentEntry is linked to a version of a Folder through a
HasMember Association. Different versions of a Folder can have different contents. This
supplement introduces rules for managing this complexity.

This supplement includes updated documentation on the use of ebRIM Associations in XDS and XDR. The existing documentation is somewhat understated. The update transaction makes heavy use of SubmissionSet Associations (Associations anchored at one end by the SubmissionSet object) to trigger some types of updates.

205 This supplement was written in parallel with the On-Demand Documents supplement. The two supplements share some common sections and terminology. The On-Demand Documents supplement depends on facilities offered by Metadata Update.

Some discussion and example material regarding the Update Document Set and Delete Document Set transactions are available on the IHE Wiki at:
210 http://wiki.ihe.net/index.php?title=Metadata_Update.

The formal example collection is available at ftp://ftp.ihe.net/TF_Implementation_Material/ITI/examples/MU.

Open Issues and Questions

Closed Issues
215

Most of the older Closed Issues have been moved to the wiki at:
http://wiki.ihe.net/index.php?title=Metadata_Update#Closed_Issues

MV012: Should the comment attribute on SubmitObjectsRequest/UpdateObjectsRequest be
220 *allowed or disallowed? No comment by IHE*

MV015: There is currently no way to issue a Stored Query asking for only online documents. Add a query parameter to FindDocuments? New query parameter not added

MV031: Metadata Update and Delete will both require significant authentication/authorization challenges. Should these be mandated by this profile or left to be decided upon by developers?
225 *Recommendations made in security sections. No mandates made.*

MV033: Since there is a significant amount of shared content and intertwined concepts should the Metadata Update and Deferred Document and Dynamically Created Content (now the On-Demand Documents) supplements be combined just for ease of understanding by the implementer? Resolution: Content was not combined.

230 *MV034: The Document Administrator can generate two transactions: Update Document Set and Delete Document Set. Both are optional but a Document Administrator must declare at least one. Is this correct or should the Document Administrator simply be required to support both transactions? Resolution: This is acceptable.*

MV035: Should XCA declare options for support of metadata versions in the query response?
235 *This could be just a required Gateway feature, especially when working within an XDS Affinity Domain where the capability exists independent of the Gateway. Another alternative might be to*

expand the option defined in the Deferred Document and dynamically generated content to include the capability of metadata versioning (rather than having two little options as is written now). Could rename the option in the other supplement to be something like "Enhanced

240   *support" and lump all this little stuff together under that. Resolution: CP 531, if passed, will enable the capabilities without the need for an option.*

MV037: An update of a DocumentEntry or Folder requires that the existing version have a non-Deprecated status. If the existing version is Deprecated, a separate operation must be used to first label it Approved (…) and then a separate operation (in a separate submission) must be

245   *used to submit the update since updates cannot be applied to deprecated objects. Should this be made easier? Resolution: restriction has been removed.*

MV037: The introduction of metadata versions makes the use of the uniqueID parameter in many Stored Queries awkward since the uniqueID represents all versions of the metadata object. Many Stored Queries, GetFolderAndContents for example, could return multiple versions of the

250   *selected Folder and the contents for each of these versions. But, if called with the entryUUID parameter the return semantics do not change. For now I have added footnotes to alert developers.*

MV038: Should ITI establish rules for the deletion of metadata? Example is a submission of a DocumentEntry with its attendant SubmissionSet and HasMember association. The minimal

255   *deletion of the DocumentEntry must delete the association as well. This leaves a SubmissionSet with no contents. Should the profile require the deletion of the SubmissionSet as well? Obviously more complicated examples exist. Resolution: IHE has not added any restrictions to be base standard. The amount of deletion is not restricted.*

MV041: Should Metadata Update apply to XDM? How? Resolution: No

260   *MV042: A public comment made issue of the lack of a defined way to request a deletion from the Document Repository. This has been recorded as Change Proposal 533.*

General Introduction

265 *Update the following appendices to the General Introduction as indicated below. Note that these are not appendices to Volume 1.*

Appendix A – Actor Summary Definitions

Add the following actors to the IHE Technical Frameworks General Introduction list of actors:

Actor	Definition
Document Administrator	The Document Administrator supports metadata update by issuing the Update Document Set transaction [ITI-57] and Delete Document Set [ITI-62] transaction to the Document Registry.

270 ## Appendix B – Transaction Summary Definitions

Add the following transactions to the IHE Technical Frameworks General Introduction list of Transactions:

Transaction	Definition
Update Document Set [ITI-57]	The Update Document Set transaction is used by the Document Administrator to issue metadata updates to the Document Registry
Delete Document Set [ITI-62]	The Delete Document Set transaction is used by the Document Administrator to request metadata deletions from the Document Registry

Glossary

275

Add the following glossary terms to the IHE Technical Frameworks General Introduction Glossary:

280 No new glossary entries.

Volume 1 – Integration Profiles

1.7 History of Annual Changes

285

Add the following bullet to the end of the bullet list in Section 1.7

- Update the XDS and XDR profiles to add support for metadata update and deletion.

2.1 Dependencies among Integration Profiles

No new dependencies are introduced with this supplement.

290

10.1 Actors/ Transactions

Update the following diagram to add the Document Administrator and the two new transactions. Note: this version of the diagram includes updates made for the On-Demand Documents supplement since it is difficult to reconcile change tracking inside diagrams. This version of the diagram takes precedence over the one in the On-Demand Documents supplement.

295
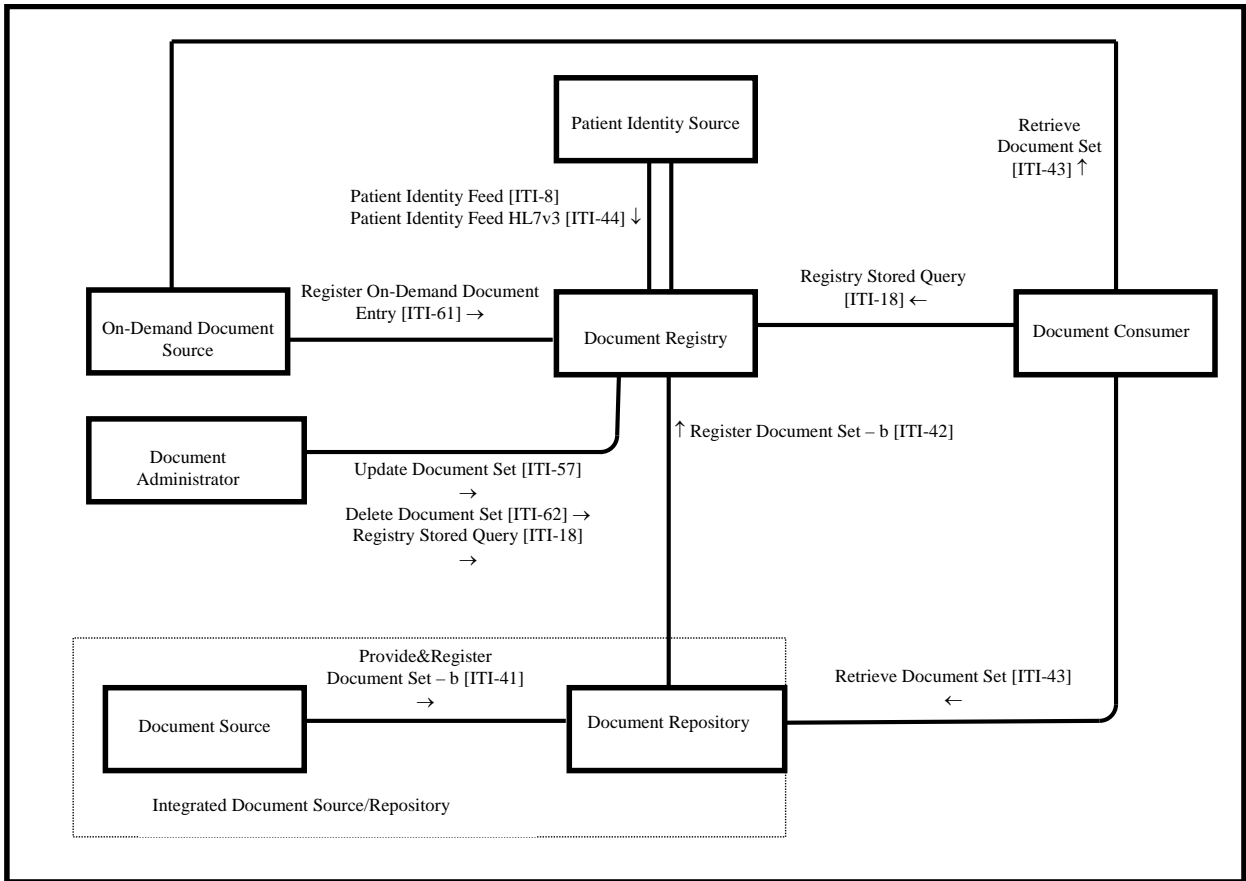
Figure 10.1-1b: Cross-Enterprise Document Sharing –b (XDS.b) Diagram

Update the following table to add the new actor and transaction.

300

Table 10.1-1b: XDS.b - Actors and Transactions

Actors	Transactions	Optionality	Section
Document Consumer	Registry Stored Query [ITI-18]	R	ITI TF-2a: 3.18
	Retrieve Document Set [ITI-43]	R	ITI TF-2b: 3.43
Document Source	Provide and Register Document Set-b [ITI-41]	R	ITI TF-2b: 3.41
Document Repository	Provide and Register Document Set-b [ITI-41]	R	ITI TF-2b: 3.41
	Register Document Set-b [ITI-42]	R	ITI TF-2b: 3.42
	Retrieve Document Set [ITI-43]	R	ITI TF-2b: 3.43
Document Registry	Register Document Set-b [ITI-42]	R	ITI TF-2b: 3.42
	Registry Stored Query [ITI-18]	R	ITI TF-2a: 3.18
	Patient Identity Feed [ITI-8]	O (Note 2)	ITI TF-2a: 3.8
	Patient Identity Feed HL7v3 [ITI-44]	O (Note 2)	ITI TF-2b: 3.44

Actors	Transactions	Optionality	Section
	Update Document Set [ITI-57]	**O**	**ITI TF-2b: 3.57**
	Delete Document Set [ITI-62]	**O**	**ITI TF-2b: 3.62**
Integrated Document Source/Repository	Register Document Set-b [ITI-42]	R	ITI TF-2b: 3.42
	Retrieve Document Set [ITI-43]	R	ITI TF-2b: 3.43
Patient Identity Source	Patient Identity Feed [ITI-8]	O (Note 1,2)	ITI TF-2a: 3.8
	Patient Identity Feed HL7v3 [ITI-44]	O (Note 1,2)	ITI TF-2b :3.44
Document Administrator	**Update Document Set [ITI-57]**	**O (Note 3)**	**ITI TF-2b: 3.57**
	Delete Document Set [ITI-62]	**O (Note 3)**	**ITI TF-2b: 3.62**
	Registry Stored Query [ITI-18]	**O**	**ITI-TF-2a: 3.18**

Note 1: If Assigning Authority of Patient ID presents in the Patient Identity Feed or Patient Identity Feed HL7v3 transaction, the Patient Identity Source is required to use an OID to identify the Assigning Authority. For technical details of the assigning authority information, see ITI TF-2a: 3.8.

305 Note 2: Document Registry and Patient Identify Source shall implement at least one of Patient Identity Feed or Patient Identity Feed HL7v3.

Note 3: Document Administrator shall support at least one of Update Document Set [ITI-57] or Delete Document Set [ITI-62].

Add sections 10.1.1.8 and 10.1.2.9 and 10.1.2.10

310 ### 10.1.1.8 Document Administrator

The Document Administrator supports metadata update by issuing the Update Document Set transaction [ITI-57] and Delete Document Set [ITI-62] transaction to the Document Registry.

10.1.2.9 Update Document Set

The Update Document Set transaction is used by the Document Administrator to issue metadata
315 updates to the Document Registry.

10.1.2.10 Delete Document Set

The Delete Document Set transaction is used by the Document Administrator to request metadata deletions from the Document Registry.

10.2 XDS Integration Profile Options

320 *Update Table 10.2-1b as follows:*

Table 10.2-1b: XDS.b - Actors and Options

Actor	Options	Vol. & Section
Document Source	Document Replacement	ITI TF-1: 10.2.1
	Document Addendum	ITI TF-1: 10.2.2
	Document Transformation	ITI TF-1: 10.2.3
	Folder Management	ITI TF-1: 10.2.4

Actor	Options	Vol. & Section
	Basic Patient Privacy Enforcement	ITI TF-2b: 3.41.4.1.3.1
Document Repository	No options defined	
Document Registry (Note 2)	Patient Identity Feed (Note 1)	ITI TF-2a: 3.8
	Patient Identity Feed HL7v3 (Note 1)	ITI TF-2b: 3.44
	Document Metadata Update	**ITI TF-1: 10.2.10**
Integrated Document Source / Repository	Document Replacement	ITI TF-1: 10.2.1
	Document Addendum	ITI TF-1: 10.2.2
	Document Transformation	ITI TF-1: 10.2.3
	Folder Management	ITI TF-1: 10.2.4
	Basic Patient Privacy Enforcement	ITI TF-2b: 3.42.4.1.4.1
Document Consumer	Basic Patient Privacy Enforcement	ITI TF-2a: 3.18.4.1.3.5 ITI TF-2b: 3.43.4.1.3.1
	Basic Patient Privacy Proof	ITI TF-2a: 3.18.4.1.3.6
	Document Metadata Update	**ITI TF-1: 10.2.10**
Patient Identity Source	Patient Identity Feed (Note 1)	ITI TF-2a: 3.8
	Patient Identity Feed HL7v3 (Note 1)	ITI TF-2b: 3.44
Document Administrator	**Update Metadata** **Delete Metadata**	**ITI TF-1: 10.2.11** **ITI TF-1: 10.2.12**

Add Section 10.2.10, Section 10.2.11 and Section 10.2.12

10.2.10 Document Metadata Update Option

325

A Document Registry declares the Document Metadata Update Option when it is able to:

1. Accept metadata updates via the Update Document Set [ITI-57] transaction (see ITI TF-2b: 3.57.4.1.3 for details)

2. Expose the metadata updates via the Registry Stored Query [ITI-18] transaction (see ITI
330 TF-2a: 3.18.4.1.2.5.1 for details)

3. Delete metadata via the Delete Document Set [ITI-62] transaction

See ITI TF-2a: 3.18.4.1.2.3.5.1 for interoperability issues surrounding the Stored Query [ITI-18] transaction.

A Document Consumer declares the Document Metadata Update Option when it is able to accept
335 and process the additional metadata defined by the option when returned from a Registry Stored Query [ITI-18] transaction (see ITI TF-2a: 3.18.4.1.2.5.1 for details). This includes:

- Multiple versions of DocumentEntry and Folder objects

- LogicalID and version attributes in DocumentEntry and Folder objects

- The DocumentEntry documentAvailability attribute.

340
- The availabilityStatus attribute on Association objects

See ITI TF-2a: 3.18.4.1.2.3.5.1 for interoperability issues surrounding the Stored Query [ITI-18] transaction.

10.2.11 Update Metadata Option

A Document Administrator that supports the Update Metadata Option shall be able to generate
345   the Update Document Set [ITI-57] transaction. A Document Administrator shall support the
Update Metadata Option or the Delete Metadata Option or both.

With this option, a Document Administrator may need to use the Stored Query [ITI-18] transaction to retrieve metadata objects. The retrieved objects are then modified and resubmitted as an update.

350   ### 10.2.12 Delete Metadata Option

A Document Administrator that supports the Delete Metadata Option shall be able to generate
the Delete Document Set [ITI-62] transaction. A Document Administrator shall support the
Update Metadata Option or the Delete Metadata Option or both.

With this option, a Document Administrator may need to use the Stored Query [ITI-18]
355   transaction to retrieve metadata objects. Some reasons for this are to determine entryUUID for an
object based on its uniqueID or to discover associations that must be deleted along with a
DocumentEntry.

10.3 XDS Process Flow

10.4 General Principles

360 | *Remove the stricken text from the next to last paragraph of Section 10.4.10.1* |
|---|

An "approved" or "deprecated" XDS Document Entry may be deleted. This change is associated
with the decision to completely remove a Document from an XDS Document Repository and the
corresponding Document Entry from the XDS Document Registry. The XDS Affinity Domain
shall establish the security policies associated with Document deletion. ~~There are no~~
365   ~~transactions defined by this Integration Profile to support such operation.~~

Add Section 10.4.14

10.4.14 Metadata Update and Delete

Metadata update is the general ability to perform maintenance on registry metadata by:

370   - Updating the attributes of a metadata object (DocumentEntry and Folder) by submitting a
new version of the entire metadata object.

- Changing the status of objects (e.g., Approved vs. Deprecated) on DocumentEntry, Folder, and Association objects

375
- Adding new Association objects. The addition of Associations is necessary to repair problems with document relationships (e.g., DocumentEntry is an addendum to the wrong base DocumentEntry) or incorrect Folder membership. In both cases the old Association must be deprecated and a new Association installed.

380
- When a metadata update changes the Patient ID attribute on DocumentEntry or Folder objects it may lead to a Folder being split, some contents keeping the old Patient ID and some getting a new Patient ID. Since the Patient ID on a Folder must match that of its contents, this may require a new Folder be submitted and some of the contents moved to it. Although this addition of Folder objects may be necessary it must be done through the Register Document Set-b [ITI-42] transaction.

Key use cases include:

385
- Update patient demographics
- Update confidentiality code
- Deprecate a document without replacing it
- Delete arbitrary registry metadata
- Remove a DocumentEntry from a Folder

390
- Remove a relationship (addendum, transformation, etc.)
- Update the Patient ID

Deleting metadata from the registry is done with the Delete Document Set transaction. The rest are accomplished with the Update Document Set transaction.

A key issue in managing metadata updates is the creation and removal of the association that:

395
- Makes a DocumentEntry a member of a Folder
- Documents a relationship between two DocumentEntries (such as Addendum or Transformation)

For example, the updating of a Folder's metadata requires the submission of a new version of the Folder object. The DocumentEntries that are members of the Folder must have their membership
400 propagated from the old version of the Folder object to the new (associations reference a particular version of an object). For each member DocumentEntry, this requires the removal (deprecation) of the association linking it to the old Folder version and the submission of a new association linking it to the new Folder version. For most updates, the rules governing this association propagation are well defined and are best performed within the Document Registry
405 Actor. The Document Administrator may trigger this with a flag on the update.

An update that changes the Patient ID attribute of a Folder or DocumentEntry is more complicated. The rules for consistency of Patient ID between Folder and member

DocumentEntries require that if a DocumentEntry gets a new Patient ID then it must be removed from the Folder (assuming the Folder does not change). The association propagation rules
410   implemented by the Document Registry do not handle this type of update. The Document Administrator must calculate and submit all of the details for changes required.

The Document Administrator signals the Document Registry when it should take responsibility for association propagation through the setting of flags on key objects in the update. These metadata flags are also used to trigger other special operations in the Document Registry such
415   the updating of availabilityStatus.

The Delete Document Set transaction requests the deletion of metadata objects from the Document Registry Actor. If the transaction is accepted the objects are permanently removed and are no longer returned in the Registry Stored Query [ITI-18] transaction. The ebRS standard, and therefore this profile, offers no provision to un-delete objects or otherwise recover deleted
420   objects.

It is beyond the scope of this profile to instruct architects and developers how to safeguard their systems when using these capabilities. Strong authentication/authorization controls are an important step. The Update Document Set transaction utilizes the SubmissionSet object to document in metadata the time, scope, and source of all updates. Additional information is
425   available through the ATNA prescribed audit logs. The Delete Document Set transaction removes arbitrary metadata objects from the registry so it must be used with extreme caution.

Add this new section ITI TF-1: 10.5.1

---

430   **10.5.1 Other Actor Grouping Rules**

10.5.1.1 Document Administrator grouped with Document Repository

The Document Administrator is grouped with the Document Repository when it is necessary to modify the online/offline status of a document in the repository. As documents in the repository are removed/restored from network accessible storage, their status can be updated in the
435   Document Registry. The XDSDocumentEntry.documentAvailability is used to label a document as online or offline. Offline indicates that the document still exists but is not currently available for retrieval.

15.1 Actors/ Transactions

Update the actors/transaction diagram in the XDR Supplement

440
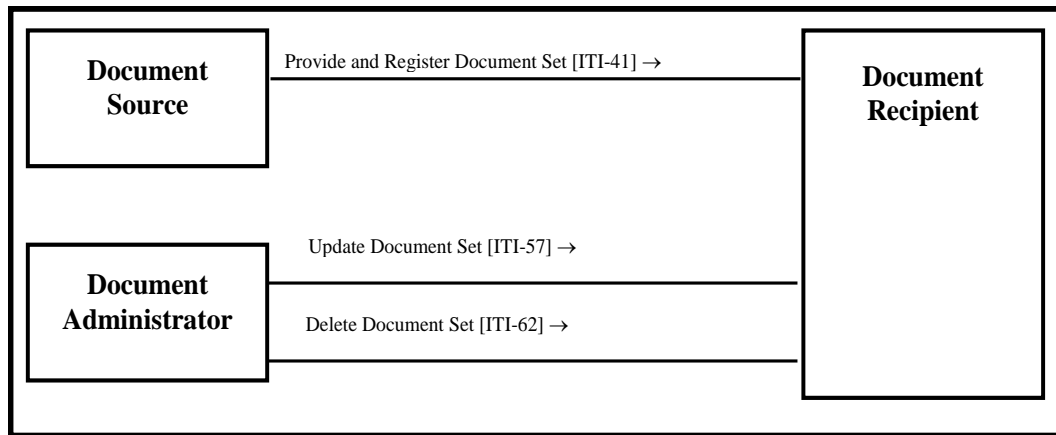
Figure 15.1-1: XDR Actor Diagram

Update the following table to add the new actors and transactions.

445

Table 15.1-1: XDR Integration Profile - Actors and Transactions

Actors	Transactions	Optionality	Section in Vol. 2
Document Source	Provide and Register Document Set-b [ITI-41]	R	ITI TF-2**b**: 3.41
Document Recipient	Provide and Register Document Set –b [ITI-41]	R	ITI TF-2**b**: 3.41
	Update Document Set [ITI-57]	**O**	**ITI TF-2b: 3.57**
	Delete Document Set [ITI-62]	**O**	**ITI TF-2b: 3.62**
Document Administrator	**Update Document Set [ITI-57]**	**O[1]**	**ITI TF-2b: 3.57**
	Delete Document Set [ITI-62]	**O[1]**	**ITI TF-2b: 3.62**

Note 1: Document Administrator shall support at least one of Update Document Set [ITI-57] or Delete Document Set [ITI-62].

450

Update the XDR Actors and Options table as follows.

455

Table 15.2-1: XDR - Actors and Options

Actor	Options	Vol. & Section
Document Source	*Multiple Document Submission*	ITI TF-1: 15.2.1
	Basic Patient Privacy Enforcement	ITI-TF-2b: 3.41.4.1.3.1
Document Recipient	*Basic Patient Privacy Enforcement*	ITI-TF-2b: 3.41.4.1.3.1
	Document Metadata Update	**ITI TF-1: 15.2.4**
Document Administrator	***Update Metadata***	**ITI TF-1: 15.2.5**
	Delete Metadata	**ITI TF-1: 15.2.6**

Add sections 15.2.4, 15.2.5, 15.2.6, 15.6 to XDR Supplement

15.2.4 Document Metadata Update Option

460 A Document Recipient that supports the Document Metadata Update Option shall be able to accept both the Update Document Set [ITI-57] transaction (see ITI TF-2b: 3.57.4.1.3 for details) and Delete Document Set [ITI-62] transaction.

15.2.5 Update Metadata Option

A Document Administrator that supports the Update Metadata Option shall be able to generate
465 the Update Document Set [ITI-57] transaction. A Document Administrator shall support the Update Metadata Option or the Delete Metadata Option or both.

15.2.6 Delete Metadata Options

A Document Administrator that supports the Delete Metadata Option shall be able to generate the Delete Document Set [ITI-62] transaction. A Document Administrator shall support the
470 Update Metadata Option or the Delete Metadata Option or both.

15.6 Metadata Update and Delete

The Document Metadata Update Option documents the use of the Update Document Set [ITI-57] transaction and Delete Document Set [ITI-62] transaction when communicating with a Document Recipient. Section ITI TF-1: 10.4.14 documents general principles of these
475 transactions that also apply to their use in XDR. An important difference is the lack of a query transaction between a Document Source and a Document Recipient. To enable the future use of update and delete, the Document Source will have to maintain information about the transmissions to permit the proper coding of future updates and deletes. To use these transactions the Document Source shall:

480 • Assign UUID format values to all entryUUID attributes transmitted in a Provide and Register-b [ITI-41] transaction

• Retain these entryUUID values for use in updates and deletes

485

- Retain version numbers for all transmitted versioned objects so the proper version information can be inserted in updates. Version numbers are never assigned as part of a metadata submission, but updates are required to include the version of the object being updated.

- In general retain knowledge of the metadata sent so that proper updates can be generated.

490 # Volume 2b – Transactions

Add Section 3.57

3.57 Update Document Set

495 *Many sections of this document make reference to terms introduced by the On-Demand Documents Supplement which is undergoing parallel development. Specifically, the following terms come from that document and readers are directed there for further reading:*

On-Demand (new type of DocumentEntry with new objectType attribute value)

Stable (new, more specific name, for what has previously been just DocumentEntry)

This section corresponds to Transaction 57 of the IHE Technical Framework. Transaction 57 is used by the Document Administrator, Document Registry and Document Recipient.

500 ### 3.57.1 Scope

The Update Document Set transaction passes a collection of metadata updates from the Document Administrator to the Document Registry or Document Recipient Actor. The update contains:

- A SubmissionSet object that organizes the content of the update

505 - Zero or more updated DocumentEntry objects

- Zero or more updated Folder objects

- Zero or more other Associations that trigger other updates

Key objects in the submission are attached to the SubmissionSet object via HasMember Associations.

510 ### 3.57.2 Use Case Roles
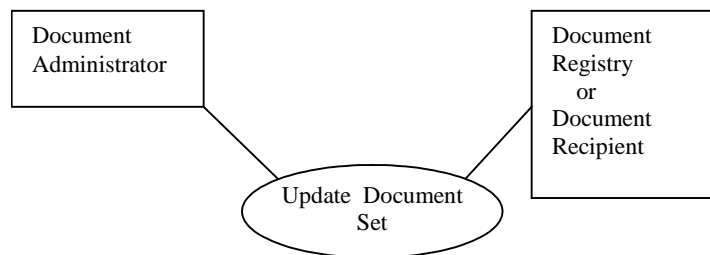
Actor: Document Administrator

Role: Issues metadata updates

515 **Actor**: Document Registry or Document Recipient

Role: Accepts metadata updates

3.57.3 Referenced Standard

Implementers of this transaction shall comply with all requirements described in ITI TF-2x: Appendix V: Web Services for IHE Transactions.

520

ebRIM	OASIS/ebXML Registry Information Model v3.0
ebRS	OASIS/ebXML Registry Services Specifications v3.0
Appendix V	ITI TF-2x: Appendix V Web Services for IHE Transactions
	Contains references to all Web Services standards and requirements of use
ITI TF-3: 4	Metadata used in Document Sharing profiles

3.57.4 Interaction Diagram
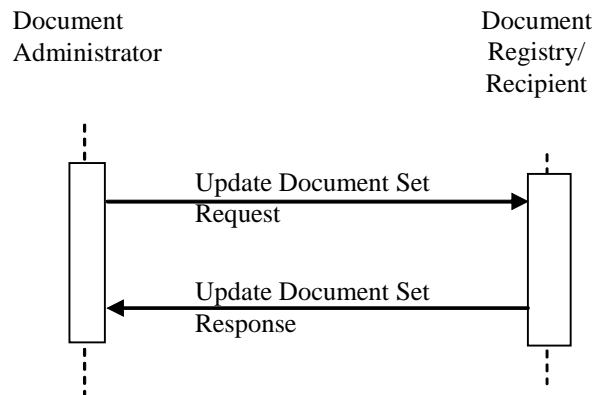
3.57.4.1 Update Document Set Request

An Update Document Set Request message provides the ability to submit the following types of
525 updates to registry metadata:

- Updated attributes for a DocumentEntry

- Updated attributes for a Folder

- Change availabilityStatus of DocumentEntry, Folder, or Association objects

- Submit new Association objects

3.57.4.1.1 Trigger Events

530

A Document Administrator needs to update the metadata attributes of existing registry objects or the Association-based linkage between them. The specific triggers for the different types of metadata updates are documented in these sections:

- 3.57.4.1.3.3.1 – Update DocumentEntry Metadata

535
- 3.57.4.1.3.3.2 – Update DocumentEntry availabilityStatus

- 3.57.4.1.3.3.3 – Update Folder Metadata

- 3.57.4.1.3.3.4 – Update Folder availabilityStatus

- 3.57.4.1.3.3.5 – Update Association availabilityStatus

- 3.57.4.1.3.3.6 – Submit Associations

3.57.4.1.2 Message Semantics

540

An Update Document Set Request message is an ebRS SubmitObjectsRequest containing ebRIM formatted metadata. The ebRIM attributes lid (logicalID in XDS) and versionName (version in XDS) are used to manage versions. See ITI TF-3: 4.1.4 for Metadata Versioning Semantics. Metadata updates are submitted as contents of a SubmissionSet.

3.57.4.1.2.1 Message Definition

545

An example Update Document Set Request message looks like:

```
<lcm:SubmitObjectsRequest>
    <rim:RegistryObjectList>
        <!-- Required SubmissionSet object -->
        <rim:RegistryPackage id="SubmissionSet"/>

        <rim:Association sourceObject="SubmissionSet"
                         targetObject="..."
                         associationType="xxx"/>
        <rim:Association sourceObject="SubmissionSet"
                         targetObject="..."
                         associationType="xxx"/>
    </rim:RegistryObjectList>
</lcm:SubmitObjectsRequest>
```

550

555

560

Note:

- The contents of the SubmissionSet object and many details of the Association objects are not shown

- Each update contained in the message includes an association linked to the
565 SubmissionSet object

- If the targetObject attribute of the association references an object already in the registry then the associationType and contained metadata (e.g., Slots nested inside the association) control the update

570
- If the targetObject attribute of the association references an object contained in the update, that object also helps control the update

Section ITI TF-3: 4.1.4 documents the semantics of metadata update.

3.57.4.1.3 Expected Actions

The Document Registry or Document Recipient parses the metadata supplied in the Update Document Set Request message and makes the updates triggered by the metadata objects in the
575 message. In general, each metadata object supplied triggers a separate metadata update. The expected actions are documented in the sub-sections below as a collection of discrete operations each triggered by a single metadata object in the message and constrained by metadata attributes contained in the message and in related objects in the registry/recipient system.

The receiving actor shall be able to manage the following attributes on DocumentEntry, Folder,
580 and Association objects:

- logicalID

- version

- availabilityStatus

which require special handling dictated by metadata update. The receiving actor shall be capable
585 of storing multiple versions of DocumentEntry and Folder metadata objects. If the receiving actor is a Document Registry then it shall make them available through the Registry Stored Query [ITI-18] transaction.

When interpreting the rules below, section ITI TF-3: 4.1.4 shall be consulted for an understanding of the semantics of metadata update.

590 **3.57.4.1.3.1 Common Rules for Metadata Update**

These common rules shall apply to all metadata update operations. They describe overall Document Registry and Document Recipient behavior with regard to metadata update. The documentation for the individual metadata update operations, found in the following sections, specialize and extend these common rules.

595 1. An Update Document Set request shall include a SubmissionSet object.

2. An Update Document Set request shall not include initial versions of DocumentEntry or Folder objects. In a request, initial versions are recognized by having entryUUID = logicalID or logicalID missing.

3. The only way to update the metadata attributes of a DocumentEntry or Folder is by
600 submitting a new version of the DocumentEntry or Folder. A special/different rule applies to the updating of the availabilityStatus attribute of a DocumentEntry or Folder.

4. An updated version of a DocumentEntry object, when included in an Update Document Set request, shall be a complete DocumentEntry object. Individual attributes cannot be submitted alone.

605   5. An updated version of a Folder object, when included in an Update Document Set request, shall be a complete Folder object. Individual attributes cannot be submitted alone.

6. Association objects cannot be updated by submitting a new version. The only attribute that can be updated is availabilityStatus. A value of Deprecated represents a historical
610   connection between two objects that is no longer current.

7. The availabilityStatus attribute on DocumentEntry, Folder, and Association objects is controlled by the Document Registry or Document Recipient Actor and cannot be altered through direct submission. A mechanism is included for requesting the registry change the value of this attribute.

615   8. The availabilityStatus of SubmissionSet Associations (sourceObject attribute references a SubmissionSet object) cannot be changed.

9. All metadata objects in an Update Document Set Request message shall have an entryUUID (id) attribute. It may be coded as a UUID or symbolic name. This attribute shall be used by the receiving actor when reporting errors.

620   10. The submission of metadata objects: DocumentEntries, Folders, Associations; in the support of metadata versioning follows the same rules as defined for the Register Document Set-b [ITI-42] transaction or Provide and Register Document Set-b [ITI-41] transaction. These rules are defined in ITI TF-3: 4.2 and 4.3.

11. A single Update Document Set transaction may contain both an Update DocumentEntry
625   Metadata operation and an Update DocumentEntry Status operation targeting the same logical DocumentEntry. When this occurs, the Document Registry or Document Recipient shall create the new DocumentEntry version and then apply the new availablityStatus value to this new version. The previous version, the one present before the processing of this transaction began, shall not be modified.

630   12. A single Update Document Set transaction may contain both an Update Folder Metadata operation and an Update Folder Status operation targeting the same logical Folder. When this occurs, the Document Registry or Document Recipient shall create the new Folder version and apply the new availablityStatus value to this new version.

13. A single Update Document Set transaction shall not contain multiple Update
635   (DocumentEntry or Folder) Metadata operations targeting the same logical object.

14. A single Update Document Set transaction shall not contain multiple Update (DocumentEntry or Folder or Association) Status operations targeting the same logical object.

15. At any point in time there shall be at most one version of a logical DocumentEntry object
640   with status Approved in the registry/recipient. If this version exists it shall always be the
       most recent version.

16. At any point in time there shall be at most one version of a logical Folder object with
    status Approved in the registry/recipient. If this version exists it shall always be the most
    recent version.

645 17. The traditional database property of atomicity shall apply to the processing of an Update
     Document Set Request message. All of the component operations of an Update
     Document Set Request message are successfully completed or no changes are made to
     the Document Registry or Document Recipient.

3.57.4.1.3.1.1 Rules for Update Planning

650

> *Editor's note*: due to the redocumentation of ITI TF-3: 4 in the ITI Technical Framework,
> *Revision 10 published in 2013, the reference below to ITI TF-3: 4.1.6.4.4.7 will get a new section*
> *number which will be determined when this supplement goes to Final Text. In this document,*
> *Section 4.1.6.4.4.7 retains this section number unchanged.*

655

The general rules found in the previous section and the rules documenting each individual update
type below are adequate for describing the individual update operations in isolation. When
related updates are specified in an Update Document Set Request a planning phase is required in
the implementation of the Document Registry or Document Recipient.

660 Related updates refers to two or more update operations that operate on overlapping registry
objects. For example, updating the metadata of a DocumentEntry that is related to second
DocumentEntry through an APND association is a well-defined operation: install the new
version of the DocumentEntry, deprecate the old version, and install a new APND association
between the new version and the related DocumentEntry. This description assumes Association
665 Propagation is enabled on the operation.

But, what if the update request also instructed the receiver to update the metadata of the
appended DocumentEntry? (Update both DocumentEntry objects bound together by the APND
association) The proper response from the receiver is to:

- Install updated version of the first DocumentEntry

670 - Install updated version of the second DocumentEntry

- Create new APND association between the new first DocumentEntry and the new second
  DocumentEntry. This assumes Association Propagation is requested for these updates
  (define below).

If the two updates of the DocumentEntries were handled independently then it is possible that:

675
- An APND association is created between the new first DocumentEntry and old second DocumentEntry because the receiver started by focusing on the update to the first DocumentEntry, then

- An APND association is created between the old first DocumentEntry and the new second DocumentEntry because the receiver moved its focus to the second
680   DocumentEntry update request and didn't notice what it did in the previous step was related.

- The result is the creation of two new associations and neither of them is correct. The correct outcome is an APND association between the two new DocumentEntry versions.

This scenario assumes that Association Propagation was enabled for both DocumentEntry
685 updates.

The result is that the Document Registry or Document Recipient Actor that is implementing Association Propagation or the Document Administrator that is formulating the update request not using Association Propagation shall:

- Recognize that two or more DocumentEntries being updated are related through
690   associations

- Recognize that a Folder and a DocumentEntry being updated are related (DocumentEntry is member of Folder)

- Install of updates to these related objects by:

    - First installing updated versions of the objects

695
    - Second performing Association Propagation to interconnect the affected objects

- Reject update requests where two related updates have different values for AssociationPropagation (see ITI TF-3: 4.1.6.4.4.7 for details)

3.57.4.1.3.2 Error Reporting

Any errors occurring in the processing of an Update Document Set Request shall cause the entire
700 transaction to fail, returning the errors detected, and causing no updates to the Document Registry. It shall be handled as a transaction, all operations succeed or no updates are made.

3.57.4.1.3.3 Metadata Operations

This section defines the metadata update operations permitted. Each operation is documented separately. Multiple metadata operations shall be accepted in a single Update Document Set
705 Request. When the Document Registry/Recipient Actor receives a submission and cannot decode the operation following the rules stated below, it shall return the error code XDSMetadataUpdateOperationError.

The technical operations that can be performed on metadata are summarized as:

- Update DocumentEntry Metadata (Stable or On-Demand) in Section 3.57.4.1.3.3.1

710
- Update DocumentEntry Status in Section 3.57.4.1.3.3.2

- Update Folder Metadata in Section 3.57.4.1.3.3.3

- Update Folder Status in Section 3.57.4.1.3.3.4

- Update Association Status in Section 3.57.4.1.3.3.5

- Submit new Association object(s) in Section 3.57.4.1.3.3.6

715 Each operation is described with the following sub-sections:

Semantics - meaning of the operation

Trigger - each metadata object in the submission, excluding the SubmissionSet object and the SubmissionSet HasMember Associations, triggers a single operation. The trigger section of each operation tells how to recognize its trigger metadata object. When a metadata object triggers
720 multiple operations, the preconditions section of the operation shall be used to resolve the conflict. If multiple or no operations are selected after considering the trigger and preconditions then the XDSMetadataUpdateOperationError shall be returned.

Preconditions - required state of objects already in the Document Registry/Recipient that are necessary to process the operation.

725 **Actions** - the actions taken by the Document Registry/Recipient Actor if the Trigger and Preconditions are met.

Association Propagation - the submission of certain types of Associations causes the registry/recipient to generate other, secondary, Associations to complete the semantic requirements of the operation.

730 **AvailabilityStatus Changes** - side effects of the operation that change the availabilityStatus attribute on objects. This does not include the normal setting of availabilityStatus to Approved for newly submitted objects. Changing the availabilityStatus on an object does not require the creation of a new version.

Patient ID Mgmt -Special handling required of the Patient ID attribute of DocumentEntry and
735 Folder objects.

3.57.4.1.3.3.1 Update DocumentEntry Metadata

3.57.4.1.3.3.1.1 Semantics

Update the metadata of a DocumentEntry object by submitting a new version of the DocumentEntry object and making this new version the current (availabilityStatus of Approved)
740 version and deprecating the previous version.

3.57.4.1.3.3.1.2 Trigger

The following rules shall be used by the receiving actor to detect an Update DocumentEntry Metadata operation in a submission:

- Submission contains a DocumentEntry object

745   1. The logicalID attribute is present in the DocumentEntry object and has a UUID formatted value.

2. The SubmissionSet to DocumentEntry HasMember Association has a Slot with name PreviousVersion. This Slot has a single value, the version number of the previous version, the one being replaced.

750   **3.57.4.1.3.3.1.3 Preconditions**

The following rules shall be used by the receiving actor to decode and validate a submission:

1. Document Registry or Document Recipient contains an existing DocumentEntry (Stable or On-Demand) instance with status of Approved.

2. Submitted DocumentEntry and existing DocumentEntry have the same values for the
755   logicalID, uniqueID, and objectType attributes.

3. The value of the PreviousVersion Slot matches the version number of the existing DocumentEntry. If all other rules succeed and this one fails, return the error code XDSMetadataVersionError.

4. The Document Registry or Document Recipient shall verify that the new DocumentEntry
760   is valid.

3.57.4.1.3.3.1.4 Actions

Store the new DocumentEntry. The version attribute is set to PreviousVersion plus one. The availabilityStatus attribute is set to the value found in the existing DocumentEntry.

3.57.4.1.3.3.1.5 Association Propagation

765

> *Editor's note: due to the redocumentation of ITI TF-3: 4 in the ITI Technical Framework, Revision 10 published in 2013, the reference below to ITI TF-3: 4.1.6.4.1 will get a new section number which will be determined when this supplement goes to Final Text. In this document, Section 4.1.6.4.1 retains this section number unchanged.*

770

Association Propagation is controlled by the AssociationPropagation slot on the SubmissionSet HasMember Association referencing the DocumentEntry object that triggered this operation. The following rules apply if this slot is missing or if its value is 'yes'. If the value is 'no' then the Document Administrator takes full responsibility for linking the updated DocumentEntry object
775   to existing metadata.

The receiving actor scans for non-deprecated HasMember Associations linking the existing DocumentEntry to a Folder. When found, a new HasMember Association is generated linking the included DocumentEntry to the same Folder.

780 The receiving actor scans for non-deprecated relationship associations (see ITI TF-3: 4.1.6.4.1) linked to the existing DocumentEntry. When found, these associations are replicated referencing the submitted DocumentEntry instead of the existing DocumentEntry. This causes the submitted DocumentEntry to inherit all the relationships from the existing DocumentEntry.

See Section 3.57.4.1.3.1.1 for additional information on Association Propagation.

3.57.4.1.3.3.1.6 AvailabilityStatus Changes

785 AvailabilityStatus of the existing DocumentEntry is changed to Deprecated.

3.57.4.1.3.3.1.7 Patient ID Management

If the patientId attribute of the existing and submitted DocumentEntries is different than the rules in Section 3.57.4.1.3.4 shall be applied. If Patient ID Reconciliation fails then the entire Update Document Set transaction shall fail. The receiving actor shall make no updates and shall return
790 the XDSPatientIDReconciliationError error code.

3.57.4.1.3.3.2 Update DocumentEntry AvailabilityStatus

3.57.4.1.3.3.2.1 Semantics

Update the availabilityStatus attribute of a DocumentEntry object by submitting an UpdateAvailabilityStatus Association anchored by the SubmissionSet.

795 ### 3.57.4.1.3.3.2.2 Trigger

The following rules shall be used by the receiving actor to detect an Update DocumentEntry Status operation in a submission:

- Submission contains an UpdateAvailabilityStatus Association:

1. The Association sourceObject references the SubmissionSet object. The targetObject is in
800 UUID format.

2. The Association contains a Slot with name of OriginalStatus containing a single value, the current namespace qualified availabilityStatus of the DocumentEntry referenced by the targetObject attribute. This Slot helps prevent race conditions when multiple Document Administrator actors are present.

805 3. The Association contains a Slot with name NewStatus containing a single value, the namespace qualified availabilityStatus value to be assigned to the DocumentEntry referenced by the targetObject attribute.

4. The value of NewStatus is a valid status for a DocumentEntry.

3.57.4.1.3.3.2.3 Preconditions

810 The following rules shall be used by the receiving actor to decode and validate a submission:

1. The Association's targetObject attribute references an existing DocumentEntry in the registry (Stable or On-Demand).

2. The DocumentEntry availabilityStatus attribute matches the value of OriginalStatus.

3. The existing DocumentEntry is the most recent instance (highest version number) for this logical DocumentEntry in the registry.

3.57.4.1.3.3.2.4 Actions

None

3.57.4.1.3.3.2.5 Association Propagation

None

3.57.4.1.3.3.2.6 AvailabilityStatus Changes

Change the availabilityStatus attribute on the existing DocumentEntry to the value found in NewStatus.

3.57.4.1.3.3.2.7 Patient ID Management

If this operation changes the status of the DocumentEntry from Deprecated to Approved then Patient ID Reconciliation Rules shall apply. See Section 3.57.4.1.3.4. If Patient ID Reconciliation fails then the entire Update Document Set transaction shall fail. The receiving actor shall make no updates and shall return the XDSPatientIDReconciliationError error code.

3.57.4.1.3.3.3 Update Folder Metadata

3.57.4.1.3.3.3.1 Semantics

Update the metadata of a Folder object by submitting a new version of the Folder object and making this new version the current (Approved) version and deprecating the previous version. At any point in time there shall be at most one version of a logical Folder object with status Approved. If this version exists it shall always be the most recent version.

3.57.4.1.3.3.3.2 Trigger

The following rules shall be used by the receiving actor to detect an Update Folder Metadata operation in a submission:

- Submission contains a Folder object

  1. This is the updated form of a Folder (entryUUID and logicalID attributes both present and have different values).

  2. The Folder logicalID attribute has a UUID formatted value. This links the update instance to the logical Folder already in the registry.

     3. The SubmissionSet to Folder HasMember Association has a Slot with name PreviousVersion. This Slot has a single value, the version number of the previous version, the one being replaced. This is used by the registry to detect update conflicts.

845 **3.57.4.1.3.3.3.3 Preconditions**

The following rules shall be used by the receiving actor to decode and validate a submission:

     1. Recipient contains an existing Folder instance with status of Approved.

     2. Submitted Folder and existing Folder have the same values for the logicalID and uniqueId attributes.

850   3. The value of the PreviousVersion Slot matches the version number of the existing Folder. If all other rules succeed and this one fails, return the error code XDSMetadataVersionError.

3.57.4.1.3.3.3.4 Actions

Store the new Folder. The version attribute is set to PreviousVersion plus one. The
855 availabilityStatus attribute is set to the value found in the existing Folder.

3.57.4.1.3.3.3.5 Association Propagation

Association Propagation is controlled by the AssociationPropagation slot on the SubmissionSet HasMember Association referencing the Folder object that triggered this operation. The following rules apply if this slot is missing or if its value is 'yes'. If the value is 'no' then the
860 Document Administrator takes full responsibility for linking the updated Folder object to existing metadata.

The receiving actor scans for non-deprecated HasMember Associations linking the existing Folder to a DocumentEntry with status Approved. When found, a new HasMember Association is generated linking the included Folder to the DocumentEntry.

865 See Section 3.57.4.1.3.1.1 for additional information on Association Propagation.

3.57.4.1.3.3.3.6 AvailabilityStatus Changes

AvailabilityStatus of the existing Folder is changed to Deprecated.

3.57.4.1.3.3.3.7 Patient ID Management

If the patientId attributes of the existing and submitted Folders are different then the rules in
870 Section 3.57.4.1.3.4 shall be applied. If Patient ID Reconciliation fails then the entire Update Document Set transaction shall fail. The receiving actor shall make no updates and shall return the XDSPatientIDReconciliationError error code.

3.57.4.1.3.3.4 Update Folder AvailabilityStatus

3.57.4.1.3.3.4.1 Semantics

875   Update the status attribute of a Folder object by submitting an UpdateAvailabilityStatus Association anchored by the SubmissionSet.

3.57.4.1.3.3.4.2 Trigger

The following rules shall be used by the receiving actor to detect an Update Folder Status operation in a submission:

880   • Submission contains a UpdateAvailabilityStatus Association:

1. The Association sourceObject references the SubmissionSet object. The targetObject is in UUID format.

2. The Association contains a Slot with name of OriginalStatus containing a single value, the current namespace qualified availabilityStatus of the Folder to be updated.
885   This Slot helps prevent race conditions when multiple Document Administrator actors are present.

3. The Association contains a Slot with name NewStatus containing a single value, the namespace qualified availabilityStatus to be assigned to the Folder.

4. The value of NewStatus is a valid status for a Folder.

890   ### 3.57.4.1.3.3.4.3 Preconditions

The following rules shall be used by the Document Registry/Recipient Actor to decode and validate a submission:

1. The UpdateAvailabilityStatus Association's targetObject attribute references an existing Folder in the registry with availabilityStatus matching the value of OriginalStatus.

895   2. The existing Folder is the most recent instance (highest version number) for this logical Folder in the registry.

3. The value of the OriginalStatus Slot matches the availabilityStatus attribute of the existing Folder.

3.57.4.1.3.3.4.4 Actions

900   None

3.57.4.1.3.3.4.5 Association Propagation

None

3.57.4.1.3.3.4.6 AvailabilityStatus Changes

Change the availabilityStatus attribute on the existing Folder to the value found in NewStatus.

905 **3.57.4.1.3.3.4.7 Patient ID Management**

If this operation changes the availabilityStatus of the Folder from Deprecated to Approved then Patient ID Reconciliation Rules shall apply. See Section 3.57.4.1.3.4. If Patient ID Reconciliation fails then the entire Update Document Set transaction shall fail. The receiving actor shall make no updates and shall return the XDSPatientIDReconciliationError error code.

910 **3.57.4.1.3.3.5 Update Association AvailabilityStatus**

3.57.4.1.3.3.5.1 Semantics

Update the availabilityStatus attribute of an Association object by submitting an UpdateAvailabilityStatus Association anchored by the SubmissionSet. This can be used to Deprecate and Undeprecate a Folder membership Association or relationship Association.

915 **3.57.4.1.3.3.5.2 Trigger**

The following rules shall be used by the receiving actor to detect an Update Association AvailabilityStatus operation in a submission:

- Submission contains an UpdateAvailabilityStatus Association.

    1. The Association sourceObject references the SubmissionSet object. The targetObject
920 is in UUID format.

    2. The Association contains a Slot with name of OriginalStatus containing a single value, the current namespace qualified availabilityStatus of the Association to be updated. This Slot helps prevent race conditions when multiple Document Administrator actors are present.

925   3. The Association contains a Slot with name NewStatus containing a single value, the namespace qualified availabilityStatus to be assigned to the Association.

3.57.4.1.3.3.5.3 Preconditions

The following rules shall be used by the receiving actor to decode and validate a submission:

    1. The UpdateAvailabilityStatus Association's targetObject attribute references an existing
930 Association in the registry.

    2. One of these two conditions shall be true:

        - The existing Association has type HasMember and the sourceObject attribute points to a Folder

        - The existing Association is a relationship type Association

935   3. The availabilityStatus attribute of the existing Association matches the value of OriginalStatus. If the existing Association has no availabilityStatus attribute then the default value of urn:oasis:names:tc:ebxml-regrep:StatusType:Approved is assumed.

3.57.4.1.3.3.5.4 Actions

None

3.57.4.1.3.3.5.5 Association Propagation

None

3.57.4.1.3.3.5.6 AvailabilityStatus Changes

Change the availabilityStatus attribute on the existing Association to the value found in NewStatus.

3.57.4.1.3.3.5.7 Patient ID Management

If this operation changes the availabilityStatus of the Association from Deprecated to Approved then Patient ID Reconciliation Rules shall apply. See Section 3.57.4.1.3.4. If Patient ID Reconciliation fails then the entire Update Document Set transaction shall fail. The receiving actor shall make no updates and shall return the XDSPatientIDReconciliationError error code.

3.57.4.1.3.3.6 Submit Associations

3.57.4.1.3.3.6.1 Semantics

It is sometimes necessary to submit arbitrary associations to repair an existing patient record. This occurs because:

- The Document Administrator does not request Association Propagation but instead includes all the necessary associations to complete the update

- The update includes changing the Patient ID on one or more objects so new associations are necessary and Association Propagation is not usable

This operation allows for that submission.

3.57.4.1.3.3.6.2 Trigger

The following rules shall be used by the receiving actor to detect a Submit Associations operation in a submission:

- Submission contains a SubmitAssociation Association.

    1. The Association sourceObject references the SubmissionSet object.

    2. The targetObject references a new association object in the submission

3.57.4.1.3.3.6.3 Preconditions

The following rules shall be used by the receiving actor to decode and validate a submission:

1. The sourceObject and targetObject attributes of the new association reference objects already in the recipient system (implies UUID format).

2. The sourceObject and targetObject shall not be deprecated.

970   3. The sourceObject and targetObject shall not reference a SubmissionSet object.

3.57.4.1.3.3.6.4 Actions

Save new association.

3.57.4.1.3.3.6.5 Association Propagation

None

975 ### 3.57.4.1.3.3.6.6 Status Changes

None

3.57.4.1.3.3.6.7 Patient ID Management

Patient ID Reconciliation rules shall be applied. If Patient ID Reconciliation fails then the entire Update Document Set transaction shall fail. The receiving actor shall make no updates and shall
980   return the XDSPatientIDReconciliationError error code.

3.57.4.1.3.4 Patient ID Reconciliation

The Update Document Set transaction can be used to update the patientId attribute of DocumentEntry and Folder objects in the receiving actor. This section documents the rules that shall be followed when updating the Patient ID attribute.

985 The following is a re-statement of the rules governing Patient IDs in metadata, including the use of versioning:

1. All SubmissionSet, Folder, and DocumentEntry objects linked by Associations shall carry the same Patient ID.

2. The only exception is linking a DocumentEntry to a Submission 'by reference' as
990   documented in ITI TF-3: 4.2.2.1.1.

These rules are not relaxed to accommodate Metadata Update. They are however extended to cover consistency between versions of a metadata object:

3. Two versions of a metadata object: DocumentEntry or Folder, are not required to carry the same Patient ID.

995 Rule #1 above is refined:

4. All SubmissionSet, approved Folder, and approved DocumentEntry objects linked by approved Associations shall carry the same Patient ID. No Patient ID consistency is expected across associations with availabiltyStatus of Deprecated or when one of the sourceObject or targetObject referenced objects has availabilityStatus of Deprecated.

1000   5. The Update Objects Request transaction is an atomic operation:

   • At the end of the transaction the above Patient ID rules shall be enforced.

- During the processing of any single element of the update, metadata will exist in the registry that does not follow the Patient ID rules.

1005
- If any part of the transaction fails, it shall leave the registry in the state it was at the beginning of the transaction.

Automatic association propagation, as defined in each metadata update operation section, is the receiver (Document Registry or Document Recipient) taking responsibility for deprecating old and generating new associations necessary for maintaining relationships in metadata with the addition of new object versions. These rules are adequate as long as the Patient ID is not

1010 updated. When the Patient ID is being updated one of two scenarios is likely:

- The Patient ID is wrong and all metadata for this Patient ID is to be updated

- A record is being split, some records keeping the original Patient ID and some getting the new one.

In either case, association propagation is not the right approach. Instead a two-step process is

1015 necessary:

1. Updates made to all (or some) related DocumentEntry and Folder objects changing the Patient ID.

2. Associations submitted to re-link all (or some) of the new DocumentEntry and Folder versions.

1020 When all objects for a Patient ID are affected, all of the involved DocumentEntry and Folder objects are updated with the new Patient ID. New associations are then installed to link the new versions into the same relationships. The old DocumentEntry and Folder objects are deprecated.

If the patient record is being split (only some of the objects get a new Patient ID) the same basic rules apply (but to only some of the objects) and there is also the possibility of document

1025 relationships and folder memberships being split and recombined to sort out the correct Patient ID assignment. The following operations are required and human supervision is likely necessary:

- Assign DocumentEntry a new Patient ID by submitting a new version of the DocumentEntry

- Deprecate relationship association between two DocumentEntries that will have different

1030 Patient IDs. See Rule # 4 in Section 3.57.4.1.3.4.

- Install new relationship association. This may or may not be possible based on the documents available. Example: an original document and its replacement end up with different Patient IDs.

- Assign Folder a new Patient ID by submitting a new version.

1035
- Create a new Folder with a new Patient ID to hold part of existing Folder contents that get a new Patient ID

- Remove DocumentEntries from a Folder by deprecating the HasMember association

    •   Add them to a new Folder by installing a new HasMember association

3.57.4.1.3.4.1 Unconnected DocumentEntry Example

1040   Updating the Patient ID on a DocumentEntry with no relationships or Folder memberships is a simple case. A new version of the DocumentEntry with the new Patient ID is submitted. The update causes the prior version to be deprecated by the receiving actor. The SubmissionSet to DocumentEntry association (original DocumentEntry) does not require deprecation because of rule #4 above.

1045   ### 3.57.4.1.3.4.2 Simple Relationship Example

Updating the Patient ID on a DocumentEntry that is an amendment for another DocumentEntry. No other relationships or Folder memberships exist. Both the original DocumentEntry and the amendment are assigned the new Patient ID. The update must include an updated version of the original DocumentEntry and the amendment DocumentEntry (with new Patient ID) and a new

1050   amend association (APND) linking the new versions. Since the receiving actor automatically deprecates both prior versions, the original APND association can be deprecated but it is not necessary. See rule #4 in Section 3.57.4.1.3.4.

3.57.4.1.3.4.3 Relationship Example with Split Patient ID

Updating the Patient ID on a DocumentEntry that has an APND relationship with another

1055   DocumentEntry but this other DocumentEntry is to retain the original Patient ID. Since the APND relationship is not allowed if the Patient IDs are different, this requires one of the following approaches to avoiding Patient ID misalignments:

    •   Deprecate the APND association and update the original DocumentEntry with a new Patient ID. The amendment DocumentEntry loses its status as an amendment but

1060   continues to exist as a document. In this case the update includes a new version of the original DocumentEntry coded with the new Patient ID and a deprecate request for the APND association.

    •   The amendment is not useful without the linkage to the original document. It is deprecated. The APND association does not require deprecation. See rule #4 in Section

1065   3.57.4.1.3.4 . In this case the update includes a new original DocumentEntry coded with the new Patient ID and a deprecate request for the amended DocumentEntry.

3.57.4.1.4 Security Considerations

Relevant XDS Affinity Domain security considerations are discussed in the XDS Security Considerations Section (see ITI TF-1: 10.7).

1070   Metadata updates are restricted to being carried only by the Update Document Set [ITI-57] transaction initiated only by the Document Administrator because the updating of a patient record involves more and different risk as compared to the submission of new elements of the patient record. We chose an independent transaction so that access controls can be isolated. Policy can then leverage the ATNA authentication to authorize updates to the patient record

1075 using the Provide and Register Document Set transaction from one node and forbid the Update Document Entry transaction from the same node.

This transaction will likely require its actors be grouped with actors in the XUA Profile by policy in most environments.

Some pertinent risks are:

1080 • Changing the confidentiality Code attribute on a DocumentEntry object allowing greater access to the underlying document

• Changing the Patient ID by stations/operators not technically competent to perform this operation

3.57.4.1.4.1 Audit Record Considerations

1085 The Update Document Set transaction is PHI-Export event, as defined in ITI TF-2a: Table 3.20.6-1 with the following exceptions.

3.57.4.1.4.1.1 Document Administrator audit message

	Field Name	Opt	Value Constraints
Event **AuditMessage/** **EventIdentification**	EventID	M	EV(110106, DCM, "Export")
	EventActionCode	M	"U" (Update)
	EventDateTime	*M*	*not specialized*
	EventOutcomeIndicator	*M*	*not specialized*
	EventTypeCode	M	EV("ITI-57", "IHE Transactions", "Update Document Set")
Source (Document Source) (1)			
Human Requestor (0..n)			
Destination (Document Repository) (1)			
Audit Source (Document Source) (1)			
Patient (1)			
SubmissionSet (1)			

Where:

	Field Name	Opt	Value Constraints
Source **AuditMessage/** **ActiveParticipant**	UserID	M	If Asynchronous Web Services Exchange is being used, the content of the <wsa:ReplyTo/> element. Otherwise, not specialized.
	AlternativeUserID	M	the process ID as used within the local operating system in the local system logs.
	UserName	*U*	*not specialized*
	UserIsRequestor	*U*	*not specialized*
	RoleIDCode	M	EV(110153, DCM, "Source")
	NetworkAccessPointTypeCode	M	"1" for machine (DNS) name, "2" for IP address
	NetworkAccessPointID	M	The machine name or IP address

1090

Human Requestor (if known) AuditMessage/ ActiveParticipant	UserID	M	Identity of the human that initiated the transaction.
	AlternativeUserID	*U*	*not specialized*
	UserName	*U*	*not specialized*
	UserIsRequestor	*U*	*not specialized*
	RoleIDCode	U	Access Control role(s) the user holds that allows this transaction.
	NetworkAccessPointTypeCode	*U*	*not specialized*
	NetworkAccessPointID	*U*	*not specialized*

Destination AuditMessage/ ActiveParticipant	UserID	M	SOAP endpoint URI.
	AlternativeUserID	*U*	*not specialized*
	UserName	*U*	*not specialized*
	UserIsRequestor	M	"false"
	RoleIDCode	M	EV(110152, DCM, "Destination")
	NetworkAccessPointTypeCode	M	"1" for machine (DNS) name, "2" for IP address
	NetworkAccessPointID	M	The machine name or IP address

Audit Source AuditMessage/ AuditSourceIdentification	*AuditSourceID*	*U*	*not specialized*
	AuditEnterpriseSiteID	*U*	*not specialized*
	AuditSourceTypeCode	*U*	*not specialized*

Patient (AuditMessage/ ParticipantObjectIdentification)	ParticipantObjectTypeCode	M	"1" (Person)
	ParticipantObjectTypeCodeRole	M	"1" (Patient)
	ParticipantObjectDataLifeCycle	*U*	*not specialized*
	ParticipantObjectIDTypeCode	M	*not specialized*
	ParticipantObjectSensitivity	*U*	*not specialized*
	ParticipantObjectID	M	The patient ID in HL7 CX format.
	ParticipantObjectName	*U*	*not specialized*
	ParticipantObjectQuery	*U*	*not specialized*
	ParticipantObjectDetail	*U*	*not specialized*

1095

1100

39

Submission Set (AuditMessage/ ParticipantObjectIdentification)	ParticipantObjectTypeCode	M	"2" (System)
	ParticipantObjectTypeCodeRole	M	"20" (job)
	ParticipantObjectDataLifeCycle	*U*	*not specialized*
	ParticipantObjectIDTypeCode	M	EV("urn:uuid:a54d6aa5-d40d-43f9-88c5-b4633d873bdd", "IHE XDS Metadata", "submission set classificationNode")
	ParticipantObjectSensitivity	*U*	*not specialized*
	ParticipantObjectID	M	The submissionSet unique ID
	ParticipantObjectName	*U*	*not specialized*
	ParticipantObjectQuery	*U*	*not specialized*
	ParticipantObjectDetail	*U*	*not specialized*

3.57.4.1.4.1.2 Document Registry or Document Recipient audit message

	Field Name	**Opt**	**Value Constraints**
Event AuditMessage/ EventIdentification	EventID	M	EV(110107, DCM, "Import")
	EventActionCode	M	"U" (Update)
	EventDateTime	*M*	*not specialized*
	EventOutcomeIndicator	*M*	*not specialized*
	EventTypeCode	M	EV("ITI-57", "IHE Transactions", "Update Document Set")
Source (Document Source) (1)			
Destination (Document Repository or Document Recipient) (1)			
Audit Source (Document Repository or Document Recipient) (1)			
Patient (1)			
SubmissionSet (1)			

1105 Where:

Source AuditMessage/ ActiveParticipant	UserID	M	If Asynchronous Web Services Exchange is being used, the content of the <wsa:ReplyTo/> element. Otherwise, not specialized.
	AlternativeUserID	*U*	*not specialized*
	UserName	*U*	*not specialized*
	UserIsRequestor	*U*	*not specialized*
	RoleIDCode	M	EV(110153, DCM, "Source")
	NetworkAccessPointTypeCode	M	"1" for machine (DNS) name, "2" for IP address
	NetworkAccessPointID	M	The machine name or IP address

Destination AuditMessage/ ActiveParticipant	UserID	M	SOAP endpoint URI
	AlternativeUserID	M	the process ID as used within the local operating system in the local system logs.
	UserName	*U*	*not specialized*
	UserIsRequestor	M	"false"
	RoleIDCode	M	EV(110152, DCM, "Destination")
	NetworkAccessPointTypeCode	M	"1" for machine (DNS) name, "2" for IP address
	NetworkAccessPointID	M	The machine name or IP address

Audit Source AuditMessage/ AuditSourceIdentificati on	*AuditSourceID*	*U*	*not specialized*
	AuditEnterpriseSiteID	*U*	*not specialized*
	AuditSourceTypeCode	*U*	*not specialized*

Patient (AuditMessage/ ParticipantObjectIde ntification)	ParticipantObjectTypeCode	M	"1" (Person)
	ParticipantObjectTypeCodeRole	M	"1" (Patient)
	ParticipantObjectDataLifeCycle	*U*	*not specialized*
	ParticipantObjectIDTypeCode	M	*not specialized*
	ParticipantObjectSensitivity	*U*	*not specialized*
	ParticipantObjectID	M	The patient ID in HL7 CX format.
	ParticipantObjectName	*U*	*not specialized*
	ParticipantObjectQuery	***U***	*not specialized*
	ParticipantObjectDetail	*U*	*not specialized*

1110

Submission Set (AuditMessage/ ParticipantObjectIde ntification)	ParticipantObjectTypeCode	M	"2" (System)
	ParticipantObjectTypeCodeRole	M	"20" (job)
	ParticipantObjectDataLifeCycle	*U*	*not specialized*
	ParticipantObjectIDTypeCode	M	EV("urn:uuid:a54d6aa5-d40d-43f9-88c5-b4633d873bdd", "IHE XDS Metadata", "submission set classificationNode")
	ParticipantObjectSensitivity	*U*	*not specialized*
	ParticipantObjectID	M	The submissionSet unique ID
	ParticipantObjectName	*U*	*not specialized*
	ParticipantObjectQuery	*U*	*not specialized*
	ParticipantObjectDetail	*U*	*not specialized*

3.57.4.2 Update Document Set Response

3.57.4.2.1 Trigger Events

After receiving the Update Document Set request, the receiving actor processes the update
1115 capturing any errors that occur. When processing is complete the response message is triggered.

3.57.4.2.2 Message Semantics

The response message is the same RegistryResponse format used by the Register Document Set-b [ITI-42] Response. The resulting status shall be Success or Failure. PartialSuccess shall not be used. Errors reported shall include the entryUUID attribute value of the object triggering the
1120 error reported in the location attribute of the RegistryError element.

3.57.4.2.3 Expected Actions

There are no expectations placed on the Document Administrator when it receives this message.

3.57.5 Protocol Requirements

The Update Document Set request and response will be transmitted using Synchronous Web Service Exchange, according to the requirements specified in ITI TF-2x: Appendix V. The protocol requirements are identical to the Register Document Set-b [ITI-42] transaction except as noted below.

- The /definitions/message/part/@element attribute of the Update Document Set Request message shall be defined as "lcm:SubmitObjectsRequest"

- The /definitions/portType/operation/input/@wsaw:Action attribute for the Register Document Set-b Request message shall be defined as "urn:ihe:iti:2010:UpdateDocumentSet"

- The /definitions/portType/operation/output/@wsaw:Action attribute for the Update Document Set Response message shall be defined as "urn:ihe:iti:2010:UpdateDocumentSetResponse"

- The /definitions/binding/operation/soap12:operation/@soapAction attribute shall be defined as "urn:ihe:iti:2010:UpdateDocumentSet"

3.57.5.1 Sample SOAP Messages

```
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
    <soapenv:Header>
        <wsa:Action mustUnderstand="1"
            xmlns:wsa="http://www.w3.org/2005/08/addressing"
            >urn:ihe:iti:2010:UpdateDocumentSetResponse</wsa:Action>
        <wsa:RelatesTo xmlns:wsa="http://www.w3.org/2005/08/addressing"
            >urn:uuid:BED84881CA1EE76D5F1278983709130</wsa:RelatesTo>
    </soapenv:Header>
    <soapenv:Body>
        <rs:RegistryResponse
          status="urn:oasis:names:tc:ebxml-regrep:ResponseStatusType:Success"
          xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"/>
    </soapenv:Body>
</soapenv:Envelope>
```

3.57.5.2 Message Examples

A collection of examples is available on the IHE FTP site   (See ITI TF-2x: Appendix W).

3.57.6 Actor Requirements

This section summarizes the responsibilities of the actors relevant to this transaction.

3.57.6.1 Document Administrator

1160 A Document Administrator that implements the Update Metadata Option shall be capable of generating at least one of the operations documented in Section 3.57.4.1.3.3.

3.57.6.2 Document Registry

A Document Registry that implements the Document Metadata Update Option shall be capable of all of the operations documented in Section 3.57.4.1.3.3.

1165 ### 3.57.6.3 Document Recipient

A Document Recipient that implements the Document Metadata Update Option shall be capable of all the operations documented in Section 3.57.4.1.3.3.

1170

1175

1180

3.62 Delete Document Set

This section corresponds to Transaction 62 of the IHE ITI Technical Framework. Transaction 62 is used by the Document Administrator, Document Recipient and Document Registry actors.

3.62.1 Scope

1185 The Delete Document Set transaction deletes arbitrary metadata objects from a Document Registry or Document Recipient based on a list of entryUUID attribute describing the objects.

3.62.2 Use Case Roles
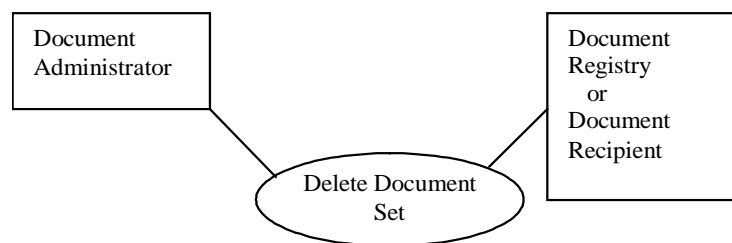
1190

Actor: Document Administrator

Role: Issues metadata delete requests to Document Registry

Actor: Document Registry or Document Recipient

Role: Accepts metadata delete requests

1195

3.62.3 Referenced Standard

Implementers of this transaction shall comply with all requirements described in ITI TF-2x: Appendix V: Web Services for IHE Transactions.

ebRIM	OASIS/ebXML Registry Information Model v3.0
ebRS	OASIS/ebXML Registry Services Specifications v3.0
Appendix V	ITI TF-2x:Appendix V Web Services for IHE Transactions
	Contains references to all Web Services standards and requirements of use
ITI TF-3:4	Metadata used in Document Sharing profiles

1200   ### 3.62.4 Interaction Diagram
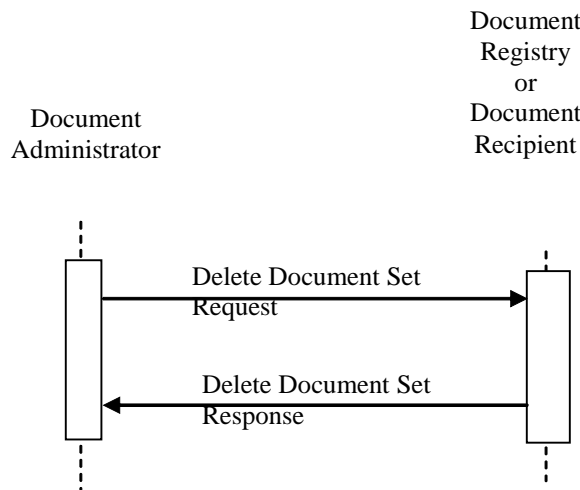
3.62.4.1 Delete Document Set Request

An ebRS RemoveObjectsRequest containing a list of ObjectRefs identifying objects to be deleted from the Document Registry. The primary documentation for this transaction is in ebRS
1205   chapter 5.6 The Remove Objects Protocol.

3.62.4.1.1 Trigger Events

The Document Administrator identifies a collection of objects to be deleted.

3.62.4.1.2 Message Semantics

1210 The Document Administrator is responsible to discover the entryUUID attribute value of the objects that are to be deleted through the Registry Stored Query [ITI-18] transaction or known history of the metadata. The entryUUID attribute values are packaged as an ObjectRef list in the request message.

Since a DocumentEntry always has at least a HasMember association to its SubmissionSet and an object cannot be deleted while association objects still reference it, to delete the
1215 DocumentEntry, the minimal deletion request is for the DocumentEntry and Association objects.

The ebRS 3.0 RemoveObjectsRequest message shall be used. The following restrictions shall be followed:

1. The AdhocQuery parameter shall not be specified. Deletions are specified only by the ObjectRefList.

1220 2. The deletionScope parameter shall not be specified.

3.62.4.1.3 Expected Actions

The Document Registry or Document Recipient Actor deletes the objects indicated in the message. The two key errors are:

- UnresolvedReferenceException – an entryUUID passed does not exist in the recipient
1225 system

- ReferencesExistException – an object to be removed is referenced by other metadata in the recipient system. An example of this is the attempted deletion of a DocumentEntry that still has an APND association referencing it. One fix for this problem is to include the entryUUID of the APND association in the delete request.

1230 Additional documentation is available in ebRS 3.0 Section 5.6.1.4.

3.62.4.2 Delete Document Set Response

An ebRS RegistryResponse message returning overall status and optionally carrying errors back to the requestor.

3.62.4.2.1 Trigger Events

1235 Processing of Delete Document Set Request is complete.

3.62.4.2.2 Message Semantics

The operation status and optional error list is returned in a RegistryResponse message. The two key errors are:

- UnresolvedReferenceException – an entryUUID passed does not exist in the recipient
1240 system

- ReferencesExistException – an object to be removed is referenced by other metadata in
  the recipient system. An example of this is the attempted deletion of a DocumentEntry
  that still has an APND association referencing it. One fix for this problem is to include
  the entryUUID of the APND association in the delete request.

1245 - Additional documentation is available in ebRS 3.0 Section 5.6.1.4.

- Details on how to construct a RegistryResponse are discussed in ITI TF-3: 4.2.4.

3.62.4.2.3 Expected Actions

None

3.62.4.2.4 Security Considerations

1250 TBD

3.62.5 Protocol Requirements

The Delete Document Set request and response will be transmitted using Synchronous Web
Service Exchange, according to the requirements specified in ITI TF-2x: Appendix V. The
protocol requirements are identical to the Register Document Set-b [ITI-42] transaction except as
1255 noted below.

- The /definitions/message/part/@element attribute of the Delete Document Set Request
  message shall be defined as "lcm:RemoveObjectsRequest"

- The /definitions/portType/operation/input/@wsaw:Action attribute for the Delete
  Document Set Request message shall be defined as
1260 "urn:ihe:iti:2010:DeleteDocumentSet"

- The /definitions/portType/operation/output/@wsaw:Action attribute for the Delete
  Document Set Response message shall be defined as
  "urn:ihe:iti:2010:DeleteDocumentSetResponse"

- The /definitions/binding/operation/soap12:operation/@soapAction attribute shall be
1265 defined as"urn:ihe:iti:2010:DeleteDocumentSet"

3.62.5.1 Sample Messages

3.62.5.1.1 Request

1270

46

```
<lcm:RemoveObjectsRequest
   xmlns:lcm="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0">
    <ObjectRefList xmlns="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0">
        <ObjectRef id="urn:uuid:b2632452-1de7-480d-94b1-c2074d79c871"/>
        <ObjectRef id="urn:uuid:b2632df2-1de7-480d-1045-c2074d79aabd"/>
        <ObjectRef id="urn:uuid:b2632132-1de7-480d-88bd-c2074d79009b"/>
        <ObjectRef id="urn:uuid:b2632cd2-1de7-480d-108c-c2074d797777"/>
    </ObjectRefList>
</lcm:RemoveObjectsRequest>
```

1275

1280

3.62.5.1.2 Response

```
<rs:RegistryResponse xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
    status="urn:oasis:names:tc:ebxml-regrep:ResponseStatusType:Success"/>
```

1285

3.62.5.1.3 Message Examples

A collection of examples are available on the IHE FTP site (see ITI TF-2x: Appendix W).

3.62.6 Actor Requirements

1290 This section summarizes the responsibilities of the actors relevant to this transaction.

3.62.6.1 Document Administrator

An implementation of the Document Administrator that implements the Delete Metadata Option shall be capable of generating the Delete Document Set transaction.

3.62.6.2 Document Registry

1295 An implementation of the Document Registry that implements the Document Metadata Update Option shall be capable of processing the Delete Document Set transaction.

3.62.6.3 Document Recipient

An implementation of the Document Recipient that implements the Document Metadata Update Option shall be capable of processing the Delete Document Set transaction.

1300

Update to 3.18 Registry Stored Query transaction

Update to ITI Vol 2a: 3.18.2 Use Case Roles

Diagram includes updates (added Document Administrator)

1305 **3.18.2 Use Case Roles**
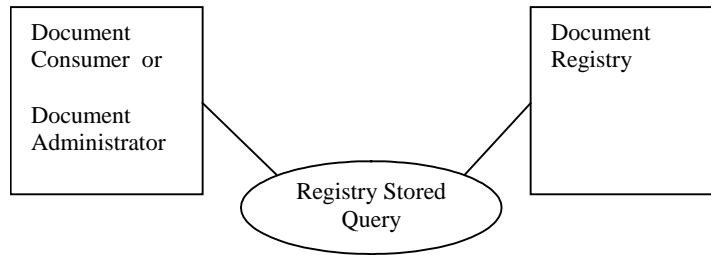
Actor: Document Consumer **or Document Administrator**

Role: Requests a query by identifier (UUID), and passes parameters to the query. A parameter controlling the format of the returned data is passed, it selects either object references or full
1310 objects.

Actor: Document Registry

Role: Services the query using its stored definitions of the queries defined for XDS

Update to 3.18 Registry Stored Query transaction

1315 *Update to ITI Vol 2a: 3.18.4 Interaction Diagram*

3.18.4 Interaction Diagram
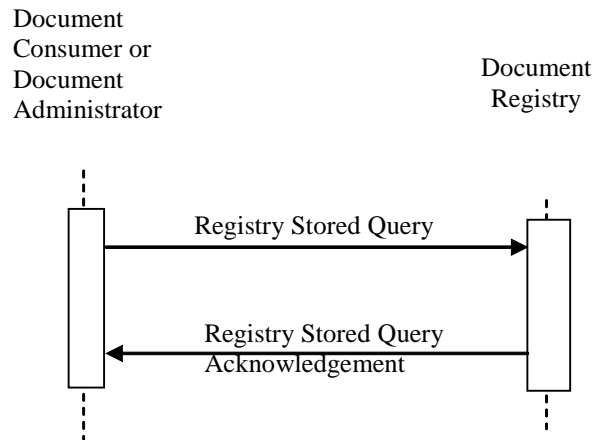
3.18 Registry Stored Query

1320

Add this section. Note the section number is odd. No extra numbering space was left in this section so I have to squeeze the section in.

3.18.4.1.2.3.5.1 Compatibility Issues

1325 **3.18.4.1.2.3.5.1.1 MetadataLevel parameter**

As the XDS Metadata Model evolves over time there is a need to protect Document Consumer implementations from incompatible changes to the Stored Query transaction so that interoperability is not degraded by the installation of more modern Document Registry implementations alongside older Document Consumer implementations.

1330 The introduction of metadata update functionality is an incompatible update to the XDS Metadata Model. The following new capabilities are likely to disrupt Document Consumer operation if it is not in some way protected:

- Labeling of a document as unavailable for retrieval (XDSDocumentEntry.documentAvailability attribute) – until the introduction of the
1335 optional Update Document Set [ITI-57] transaction, all documents indexed in the Document Registry were expected to be retrievable

- Deprecation of association objects – until the introduction of the optional Update Document Set [ITI-57] transaction, the status attribute on the association object (availabilityStatus attribute as documented by XDS) was not used by XDS. It can now be
1340 used to deprecate an association, transforming that association from an active part of the patient record into a historical part of the patient record.

- Deprecation of Folder objects – while some Stored Queries have a place to specify the availabilityStatus of Folder objects, others do not.

These three capabilities introduce metadata patterns that a pre-metadata update Document
1345 Consumer could incorrectly interpret parts of the medical record as expressed in XDS metadata. To guard against this, a new optional Stored Query parameter, $MetadataLevel, is introduced to most Stored Queries to allow the Document Consumer to specify which version of the XDS Metadata Model it can interpret. Queries where it is not available do not interact with the metadata update capabilities.

1350 The default value of $MetadataLevel is 1 (one), if it is not present as a parameter then is values is assumed to be 1, indicating that the following metadata patterns shall not be returned in the Stored Query response:

- DocumentEntry objects containing the documentAvailability attribute with values other than `urn:ihe:iti:2010:DocumentAvailability:Online`.

1355 - Association objects with availabilityStatus attribute with values other than `urn:oasis:names:tc:ebxml-regrep:StatusType:Approved`.

- Folder objects with availabilityStatus attribute set to the value `urn:oasis:names:tc:ebxml-regrep:StatusType:Approved`.

Individual Stored Queries may define more specific behavior if appropriate.

1360 Stored Query requests coded with $MetadataLevel value of 2 shall return without the above restrictions as is appropriate for the individual Stored Query and its other parameters.

The $MetadataLevel parameter shall be coded as a number. See Section 3.18.4.1.2.3.5.

3.18.4.1.2.3.5.1.2 Stored Query parameters added by Metadata Update Option

The following parameters are added by the Metadata Update Option.

1365

Stored Query	Added parameters
FindDocuments	$XDSDocumentEntryDocumentAvailability
	$MetadataLevel
FindFolders	$MetadataLevel
GetAll	$XDSAssociationStatus
	$MetadataLevel
GetDocuments	$XDSDocumentEntryLogicalID
	$MetadataLevel
GetFolders	$XDSFolderLogicalID
	$MetadataLevel
GetAssociations	$XDSAssociationStatus
	$MetadataLevel
GetDocumentsAndAssociations	$XDSAssociationStatus
	$MetadataLevel
GetSubmissionSets	$MetadataLevel
GetSubmissionSetAndContents	$MetadataLevel
GetFolderAndContents	$XDSAssociationStatus
	$MetadataLevel
GetFoldersForDocument	$XDSAssociationStatus
	$MetadataLevel
GetRelatedDocuments	$XDSAssociationStatus
	$MetadataLevel

3.18.4.1.2.3.5.1.3 Metadata Update Option compatibility

This section documents compatibility issues that arise between the Document Registry and Document Consumer with the implementation of the Metadata Update Option.

1370 Metadata Update not implemented by Document Registry or Document Consumer

- No issues

Metadata Update implemented by both Document Registry and Document Consumer

- No Issues

Metadata Update Option implemented by Document Consumer but not Document Registry

1375
- $MetadataLevel parameter specified in query, ignored by Document Registry

- Other new parameters specified in query, ignored by Document Registry

- In GetDocuments and GetFolders queries, the logicalID parameter can be specified. The Document Registry will not recognize it. The registry will recognize that neither the UUID nor uniqueID parameter is provided. An error will be returned. This may be the
1380 first indication the Document Consumer has that the Document Registry is not Metadata Update enabled. The Document Consumer must recover gracefully.

- Requests for deprecated Folders and Associations will not return the expected contents.

- The $XDSDocumentEntryDocumentAvailabilty parameter to FindDocuments will be ignored by the registry. The will cause no surprises since such a registry will not maintain
1385 the documentAvailability attribute.

Metadata Update Option implemented by the Document Registry but not the Document Consumer

- The $MetadataLevel parameter will be 1 or not present. The Document Consumer may set $MetadataLevel to 1 if it wishes the query return to follow the no Metadata Update
1390 rules.

- The Document Registry shall return results consistent with pre-Metadata Update rules. See Section 3.18.4.1.2.3.5.1.1 for details.

> *Update to FindDocuments section – add documentation to $XDSDocumentEntryStatus*
1395 > *parameter*

3.18.4.1.2.3.7.1 FindDocuments

Parameter Name	Attribute	Opt	Mult
$XDSDocumentEntryStatus[5]	XDSDocumentE	R	M

Parameter Name	Attribute	Opt	Mult
	ntry.status		
$XDSDocumentEntryDocumentAvailability	**XDSDocumentEntry.documentAvailability**	**O**	**M**
$MetadataLevel[6]	**None**	**O**	**--**

[5]If the Document Registry supports the Document Metadata Update Option and this parameter includes the value urn:oasis:names:tc:ebxml-regrep:StatusType:Deprecated then the response may include multiple versions of a DocumentEntry object. See ITI TF-3: 4.1.4 for guidance on interpreting these results.

[6]See Section 3.18.4.1.2.3.5.1

Update to FindFolders section – add documentation to $XDSFolderStatus parameter

3.18.4.1.2.3.7.2 FindFolders

Parameter Name	Attribute	Opt	Mult
$XDSFolderStatus[4]	XDSFolder.status	R	M
$MetadataLevel[5]	**None**	**O**	**--**

[4]If the Document Registry supports the Document Metadata Update Option and this parameter includes the value urn:oasis:names:tc:ebxml-regrep:StatusType:Deprecated then the response may include multiple versions of a Folder object. See ITI TF-3: 4.1.4 for guidance on interpreting these results.

[5] The parameter setting of $MetadataLevel equals 1 shall not inhibit the return of non-Approved Folders. This is controlled only by $XDSFolderStatus. See Section 3.18.4.1.2.3.5.1.

Update to GetAll section – add documentation to $XDSDocumentEntry and $XDSFolderStatus parameters

3.18.4.1.2.3.7.3 GetAll

Get all registry content for a patient given the indicated status, format codes, and confidentiality codes.

Returns:

- XDSSubmissionSet, XDSDocumentEntry, and XDSFolder objects with patientId attribute matching $patientId parameter

1425
- Association objects with sourceObject or targetObject attribute matching one of the above objects **and availabilityStatus matching one of the values in the $XDSAssociationStatus parameter.**

Parameter Name	Attribute	Opt	Mult
$XDSDocumentEntryStatus[3]	XDSDocumentEntry.status	R	M
$XDSFolderStatus[3]	XDSFolder.status	R	M
$XDSAssociationStatus	**XDSAssociation.status**	**O**	**M**
$MetadataLevel[4]	**None**	**O**	**--**

[3]If the Document Registry supports the Document Metadata Update Option and this parameter includes the value urn:oasis:names:tc:ebxml-regrep:StatusType:Deprecated
1430 **then the response may include multiple versions of a DocumentEntry/Folder object. See ITI TF-3: 4.1.4 for guidance on interpreting these results.**

[4] The parameter setting of $MetadataLevel equals 1 shall not inhibit the return of non-Approved Folders. This is controlled only by $XDSFolderStatus. See Section 3.18.4.1.2.3.5.1.

1435

Update to Vol 2a:– Add parameter to GetDocuments Stored Query to allow retrieval of DocumentEntry objects by their logicalID.

3.18.4.1.2.3.7.5 GetDocuments

1440 Retrieve a collection of XDSDocumentEntry objects. XDSDocumentEntry objects are selected either by their entryUUID, ~~or~~ uniqueID**, or logicalID** attribute.

Parameter Name	Attribute	Opt	Mult
$XDSDocumentEntryEntryUUID	XDSDocumentEntry. entryUUID	O[1]	M
$XDSDocumentEntryUniqueId[4]	XDSDocumentEntry. uniqueId	O[1]	M
$XDSDocumentEntryLogicalID[3]	**XDSDocumentEntry.logicalID**	**O[1]**	**M**
$homeCommunityId	None	O[2]	--
$MetadataLevel[5]	**None**	**O**	**--**

[1]Either $XDSDocumentEntry**.**EntryUUID or $XDSDocumentEntry**.**UniqueId **or**
1445   **$XDSDocumentEntry.logicalID** shall be specified. This transaction shall return an error if ~~both~~
more than one of these parameters ~~are~~ **is** specified.

[2]The homeCommunityId value is specified as the home attribute on the AdhocQuery element of
the query request, as in: <AdhocQuery id=″…″ home=″urn:oid:1.2.3″ … >. Document
Consumer actors shall specify the homeCommunityId value if they received a value for this
1450   attribute as part of the previous Registry Stored Query response entry, which contained the
specified ~~EntryUUID or UniqueID~~ **entryUUID, uniqueId or logicalID**.

See ~~ITI TF-2a:~~ **Section** 3.18.4.1.2.3.8 for more details.

[3]**Retrieving a DocumentEntry by its logicalID attribute returns the logical DocumentEntry,**
which includes all versions of that DocumentEntry object. See ITI TF-3: 4.1.4 for guidance
1455   **on interpreting these results. If the Document Registry does not support the Document**
Metadata Update Option this parameter is not allowed.

[4]**Specifying this parameter in a query to a Document Registry that supports the Document**
Metadata Update Option may return multiple DocumentEntry objects each representing a
different version of the metadata. See ITI TF-3: 4.1.4 for guidance on interpreting these
1460   **results.**

[5]**See Section 3.18.4.1.2.3.5.1**

Update to Vol 2a:– Add parameter to GetFolders Stored Query to allow retrieval of
DocumentEntry objects by their logicalID.

1465   **3.18.4.1.2.3.7.6 GetFolders**

Parameter Name	Attribute	Opt	Mult
$XDSFolderEntryUUID	XDSFolder. entryUUID	O[1]	M
$XDSFolderUniqueId[4]	XDSFolder. uniqueId	O[1]	M
$XDSFolderLogicalID[3]	**XDSFolder.logicalID**	**O[1]**	**M**
$homeCommunityId	None	O[2]	--
$MetadataLevel[5]	**None**	**O**	**--**

[1]Either $XDSFolder**.**EntryUUID or $XDSFolder**.u~~U~~**niqueId **or $XDSFolder.logicalID** shall be
specified. This transaction shall return an error if ~~both~~ more than one of these parameters are
1470   specified.

[2]The homeCommunityId value is specified as the home attribute on the AdhocQuery element of
the query request, as in: <AdhocQuery id=″…″ home=″urn:oid:1.2.3″ … >. Document

---

Consumer actors shall specify the homeCommunityId value if they received a value for this
attribute as part of the previous Registry Stored Query response entry, which contained the
1475   specified ~~EntryUUID or UniqueID~~ **entryUUID, uniqueId or logicalID**.

See ~~ITI TF-2a:~~**Section** 3.18.4.1.2.3.8 for more details.

**[3]Retrieving a Folder by its logicalID attribute returns the logical Folder, which includes all
versions of that Folder object. See ITI TF-3: 4.1.4 for guidance on interpreting these
results. If the Document Registry does not support the Document Metadata Update Option**
1480   **this parameter is not allowed.**

**[4]Specifying this parameter in a query to a Document Registry that supports the Document
Metadata Update Option may return multiple Folder objects each representing a different
version of the metadata. See ITI TF-3: 4.1.4 for guidance on interpreting these results. Also
see Section** 3.18.4.1.2.3.5.1 for considerations regarding $MetadataLevel.

1485   **[5]See Section 3.18.4.1.2.3.5.1**

Update to Vol 2a:– Add parameter to GetAssociations Stored Query.

3.18.4.1.2.3.7.7 GetAssociations

Parameter Name	Attribute	Opt	Mult
$XDSAssociationStatus	**Association.status**	**O**	**M**
$MetadataLevel[3]	**None**	**O**	**--**

1490   **[3]See Section 3.18.4.1.2.3.5.1**

---

*Update to Vol 2a:– Add footnote to GetDocumentsAndAssociations Stored Query. Also update
text to show sensitivity to Association.availabilityStatus*

1495   ### 3.18.4.1.2.3.7.8 GetDocumentsAndAssociations

Retrieve a collection of XDSDocumentEntry objects and the Association objects surrounding
them. XDSDocumentEntry objects are selected either by their entryUUID or uniqueId attribute.
This is the GetDocuments query and GetAssociations query combined into a single query.

Returns:

1500   • XDSDocumentEntry objects.

    • Association objects whose sourceObject or targetObject attribute matches one of the
      above objects **and availabilityStatus matching one of the values in the
      $XDSAssociationStatus parameter.**

Parameter Name	Attribute	Opt	Mult
$XDSDocumentEntryUniqueId[3]	XDSDocumentEntry.uniqueId	O[1]	M
$XDSAssociationStatus	**XDSAssociation.status**	**O**	**M**
$MetadataLevel[4]	**None**	**O**	**--**

1505 **[3]Specifying this parameter in a query to a Document Registry that supports the Document Metadata Update Option may return multiple DocumentEntry objects each representing a different version of the metadata. See ITI TF-3: 4.1.4 for guidance on interpreting these results.**

[4]See Section 3.18.4.1.2.3.5.1

1510 | *Update to Vol 2a: - Add $MetadataLevel parameter to parameter table* |

3.18.4.1.2.3.7.9 GetSubmissionSets

Parameter Name	Attribute	Opt	Mult
$MetadataLevel[2]	**None**	**O**	**--**

[2]See Section 3.18.4.1.2.3.5.1

1515 | *Update to Vol 2a: - Add $MetadataLevel parameter to parameter table* |

3.18.4.1.2.3.7.10 GetSubmissionSetAndContents

Parameter Name	Attribute	Opt	Mult
$MetadataLevel[5]	**None**	**O**	**--**

[5]See Section 3.18.4.1.2.3.5.1

1520 | *Update to Vol 2a:– Add footnote to GetFolderAndContents Stored Query. Also update text to show sensitivity to Association.availabilityStatus* |

3.18.4.1.2.3.7.11 GetFolderAndContents

1525 Retrieve an XDSFolder object and its contents. XDSFolder objects are selected either by their entryUUID or uniqueId attribute. The XDSDocumentEntry objects returned shall match one of the confidentiality codes listed if that parameter is included.

Returns:

- XDSFolder object specified in the query.

1530
- Association objects of type HasMember that have a sourceObject attribute referencing the XDSFolder object specified in the query **and availabilityStatus matching one of the values in the $XDSAssociationStatus parameter.**

- XDSDocumentEntry objects referenced by the targetObject attribute of one of the Association objects specified above.

1535

Parameter Name	Attribute	Opt	Mult
$XDSFolderUniqueId**5**	XDSFolder.uniqueId	O[1]	--
$XDSAssociationStatus	**XDSAssociation.status**	**O**	**M**
$MetadataLevel[6]	**None**	**O**	**--**

[5]Specifying this parameter in a query to a Document Registry that supports the Document Metadata Update Option may return multiple Folder objects each representing a different version of the metadata. By the rules of this query, the contents of each version will also be returned. See ITI TF-3: 4.1.4 for guidance on interpreting these results.

1540 **[6]See Section 3.18.4.1.2.3.5.1**

> *Update to Vol 2a:– Add footnote to GetFoldersForDocument Stored Query. Also update text to show sensitivity to Association.availabilityStatus*

1545

3.18.4.1.2.3.7.12 GetFoldersForDocument

Retrieve XDSFolder objects that contain the XDSDocumentEntry object provided with the query. XDSDocumentEntry objects are selected either by their entryUUID or uniqueId attribute.

Returns: XDSFolder objects that contain specified XDSDocumentEntry object. More
1550 specifically, for each Association object of type HasMember that has a targetObject attribute referencing the target XDSDocumentEntry object **and availabilityStatus matching one of the values in the $XDSAssociationStatus parameter**, return the object referenced by its sourceObject if it is of type XDSFolder.

Parameter Name	Attribute	Opt	Mult
$XDSDocumentEntryUniqueId**[3]**	XDSDocumentEntry.uniqueId	O[1]	--
$XDSAssociationStatus	**XDSAssociation.status**	**O**	**M**
$MetadataLevel[4]	**None**	**O**	**--**

1555 **[3]Specifying this parameter in a query to a Document Registry that supports the Document Metadata Update Option may return multiple DocumentEntry objects each representing a different version of the metadata. By the rules of this query, the Folders containing each version will also be returned. See ITI TF-3: 4.1.4 for guidance on interpreting these results.**

[4]See Section 3.18.4.1.2.3.5.1

1560

> *Update to Vol 2a:– Add footnote to GetRelatedDocuments Stored Query. Also update text to show sensitivity to Association.availabilityStatus*

1565 ### 3.18.4.1.2.3.7.13 GetRelatedDocuments

Retrieve XDSDocumentEntry objects that are related to the specified document via Association objects. Also return the Association objects. The specified document is designated by UUID or uniqueId. The query shall return

- Association objects where:

1570
  - The sourceObject attribute OR the targetObject attribute references the specified document

    AND

  - **The availabilityStatus matching one of the values in the $XDSAssociationStatus parameter.**

1575 - Both sourceObject attribute and targetObject attribute reference documents

  AND

  - The associationType attribute matches a value included in the $AssociationTypes parameter

Parameter Name	Attribute	Opt	Mult
$XDSDocumentEntryUniqueId**[3]**	XDSDocumentEntry.uniqueId	O[1]	--
$XDSAssociationStatus	**XDSAssociation.status**	**O**	**M**

$MetadataLevel[4]	**None**	**O**	**--**

1580 **[3]Specifying this parameter in a query to a Document Registry that supports the Document Metadata Update Option may return multiple DocumentEntry objects each representing a different version of the metadata. By the rules of this query, the results will include DocumentEntry objects related to each version of the DocumentEntry selected by its uniqueID. See ITI TF-3: 4.1.4 for guidance on interpreting these results.**

1585 **[4]See Section 3.18.4.1.2.3.5.1.**

> *Update to Registry Stored Query transaction*
>
> *Update to ITI Vol 2a: 3.18.4.1*

1590

3.18.4.1.1 Trigger Events

This message is initiated when the Document Consumer **or Document Administrator** wants to query/retrieve document metadata.

1595 > *Add this content to Vol 2b: 3.18.4.1.2.5 (currently labeled Intentionally Left Blank)*

3.18.4.1.2.5 Options

The following options affect the operation of this transaction.

3.18.4.1.2.5.1 Document Metadata Update Option

1600 **When the Document Registry declares this option it affects this transaction as follows:**

- **The logicalID and version attributes shall be returned in responses**

- **Multiple DocumentEntry objects may be returned carrying the same uniqueId attribute reflecting different versions of the object. At most one version shall have a non-Deprecated status.**

1605 - **Multiple Folder objects may be returned carrying the same uniqueId attribute reflecting different versions of the object. At most one version shall have a non-Deprecated status.**

- **Association objects returned shall include the availabilityStatus attribute**

- **Association objects returned will include Slots that reflect the update history of the**
1610 **objects they reference.**

When the Document Consumer declares this option the above considerations apply. Such a Document Consumer shall be constructed considering the following effects:

- **Objects returned with availabilityStatus of Deprecated may carry attributes which are out of date but which reflected the best known information at the time of their submission**

1615

- **Patient ID agreement is not guaranteed on Deprecated objects**

- **Relationship and Folder membership associations may be deprecated thus cancelling the relationship or membership**

1620

3.18.4.1.3 Expected Actions

The Document Consumer shall accept the metadata returned in the response and process it
1625 **according to its own rules.**

If the Document Consumer implements the Document Metadata Update Option and it is querying an actor that implements the Document Metadata Update Option then it shall:

1. **Use query options to control how much metadata is returned (control how many / which versions of an object are returned)**

1630 2. **Distinguish between multiple versions of DocumentEntry and Folder objects**

3. **Recognize deprecated Folder and Association objects**

4. **Understand the use of logicalID and version attributes**

The Document Administrator may use the contents of this transaction to guide its submission of updates and deletions. If used it shall:

1635 1. **Note the version attribute on DocumentEntry objects. This shall be submitted as the PreviousVersion attribute on the SubmissionSet to DocumentEntry HasMember association when updating the DocumentEntry metadata.**

2. **Note the value of the availabilityStatus on objects to be updated. This attribute is updated without changing the version of the object.**

1640

3.18.4.1.3.3 Sample Query Response

```
<?xml version="1.0" encoding="UTF-8"?>
<AdhocQueryResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1645     xmlns="urn:oasis:names:tc:ebxml-regrep:xsd:query:3.0"
    xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
    status="urn:oasis:names:tc:ebxml-regrep:ResponseStatusType:Success">
    <rim:RegistryObjectList>
        <rim:ExtrinsicObject xmlns:q="urn:oasis:names:tc:ebxml-
1650 regrep:xsd:query:3.0"
            xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
            id="urn:uuid:08a15a6f-5b4a-42de-8f95-89474f83abdf"
            lid="urn:uuid:08a15a6f-5b4a-42de-8f95-89474f83abdf"
mimeType="text/xml"
1655            objectType="urn:uuid:7edca82f-054d-47f2-a032-9b2a5b5186c1"
```

```
                    status="urn:oasis:names:tc:ebxml-regrep:StatusType:Approved">
              <rim:Slot name="authorInstitution">
                  <rim:ValueList>
                      <rim:Value>Some
Hospital^^^^^^^^^1.2.3.4.5.6.7.8.9.1789.45</rim:Value>
                  </rim:ValueList>
              </rim:Slot>
              <rim:Slot name="creationTime">
                  <rim:ValueList>
                      <rim:Value>200412261119</rim:Value>
                  </rim:ValueList>
              </rim:Slot>
              <rim:Slot name="hash">
                  <rim:ValueList>

<rim:Value>4cf4f82d78b5e2aac35c31bca8cb79fe6bd6a41e</rim:Value>
                  </rim:ValueList>
              </rim:Slot>
              <rim:Slot name="languageCode">
                  <rim:ValueList>
                      <rim:Value>en-us</rim:Value>
                  </rim:ValueList>
              </rim:Slot>
              <rim:Slot name="serviceStartTime">
                  <rim:ValueList>
                      <rim:Value>200412230800</rim:Value>
                  </rim:ValueList>
              </rim:Slot>
              <rim:Slot name="serviceStopTime">
                  <rim:ValueList>
                      <rim:Value>200412230801</rim:Value>
                  </rim:ValueList>
              </rim:Slot>
              <rim:Slot name="size">
                  <rim:ValueList>
                      <rim:Value>54449</rim:Value>
                  </rim:ValueList>
              </rim:Slot>
              <rim:Slot name="sourcePatientId">
                  <rim:ValueList>
                      <rim:Value>jd12323^^^&amp;1.2.3&amp;ISO</rim:Value>
                  </rim:ValueList>
              </rim:Slot>
```

```
                    <rim:Slot name="sourcePatientInfo">
1700                    <rim:ValueList>
                            <rim:Value>PID-3|pid1^^^&amp;3.4.5.6&amp;ISO</rim:Value>
                            <rim:Value>PID-5|Doe^John^^^</rim:Value>
                            <rim:Value>PID-7|19560527</rim:Value>
                            <rim:Value>PID-8|M</rim:Value>
1705                        <rim:Value>PID-11|100 Main
St^^Metropolis^Il^44130^USA</rim:Value>
                        </rim:ValueList>
                    </rim:Slot>
                    <rim:Name>
1710                    <rim:LocalizedString charset="UTF-8" value="Sample document
1" xml:lang="en-us"/>
                    </rim:Name>
                    <rim:Description/>
                    <VersionInfo versionName="1"/>
1715                <rim:Classification classificationScheme="urn:uuid:41a5887f-8865-
4c09-adf7-e362475b143a"
                        classifiedObject="urn:uuid:08a15a6f-5b4a-42de-8f95-
89474f83abdf"
                        id="urn:uuid:ac872fc0-1c6e-439f-84d1-f76770a0ccdf"
1720                    lid="urn:uuid:ac872fc0-1c6e-439f-84d1-f76770a0ccdf"
nodeRepresentation="Education"
                        objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:Classification">
                        <rim:Slot name="codingScheme">
1725                        <rim:ValueList>
                                <rim:Value>Connect-a-thon classCodes</rim:Value>
                            </rim:ValueList>
                        </rim:Slot>
                        <rim:Name>
1730                        <rim:LocalizedString charset="UTF-8" value="Education"
xml:lang="en-us"/>
                        </rim:Name>
                        <rim:Description/>
                        <VersionInfo versionName="1"/>
1735                </rim:Classification>
                    <rim:Classification classificationScheme="urn:uuid:f4f85eac-e6cb-
4883-b524-f2705394840f"
                        classifiedObject="urn:uuid:08a15a6f-5b4a-42de-8f95-
89474f83abdf"
1740                    id="urn:uuid:f1a8c8e4-3593-4777-b7e0-8b0773378705"
                        lid="urn:uuid:f1a8c8e4-3593-4777-b7e0-8b0773378705"
nodeRepresentation="C"
```

```
                         objectType="urn:oasis:names:tc:ebxml-
          regrep:ObjectType:RegistryObject:Classification">
1745                  <rim:Slot name="codingScheme">
                          <rim:ValueList>
                              <rim:Value>Connect-a-thon
          confidentialityCodes</rim:Value>
                          </rim:ValueList>
1750                  </rim:Slot>
                      <rim:Name>
                          <rim:LocalizedString charset="UTF-8" value="Celebrity"
          xml:lang="en-us"/>
                      </rim:Name>
1755                  <rim:Description/>
                      <VersionInfo versionName="1"/>
                  </rim:Classification>
                  <rim:Classification classificationScheme="urn:uuid:a09d5840-386c-
          46f2-b5ad-9c3699a4309d"
1760                  classifiedObject="urn:uuid:08a15a6f-5b4a-42de-8f95-
          89474f83abdf"
                      id="urn:uuid:b6e49c73-96c8-4058-8c95-914d83bd262a"
                      lid="urn:uuid:b6e49c73-96c8-4058-8c95-914d83bd262a"
                      nodeRepresentation="CDAR2/IHE 1.0"
1765                  objectType="urn:oasis:names:tc:ebxml-
          regrep:ObjectType:RegistryObject:Classification">
                      <rim:Slot name="codingScheme">
                          <rim:ValueList>
                              <rim:Value>Connect-a-thon formatCodes</rim:Value>
1770                      </rim:ValueList>
                      </rim:Slot>
                      <rim:Name>
                          <rim:LocalizedString charset="UTF-8" value="CDAR2/IHE
          1.0" xml:lang="en-us"/>
1775                  </rim:Name>
                      <rim:Description/>
                      <VersionInfo versionName="1"/>
                  </rim:Classification>
                  <rim:Classification classificationScheme="urn:uuid:f33fb8ac-18af-
1780          42cc-ae0e-ed0b0bdb91e1"
                      classifiedObject="urn:uuid:08a15a6f-5b4a-42de-8f95-
          89474f83abdf"
                      id="urn:uuid:61e2b376-d74a-4984-ac21-dcd0b8890f9d"
                      lid="urn:uuid:61e2b376-d74a-4984-ac21-dcd0b8890f9d"
1785          nodeRepresentation="Emergency Department"
```

```
                    objectType="urn:oasis:names:tc:ebxml-
        regrep:ObjectType:RegistryObject:Classification">
                    <rim:Slot name="codingScheme">
                        <rim:ValueList>
1790                         <rim:Value>Connect-a-thon
        healthcareFacilityTypeCodes</rim:Value>
                        </rim:ValueList>
                    </rim:Slot>
                    <rim:Name>
1795                        <rim:LocalizedString charset="UTF-8" value="Assisted
        Living" xml:lang="en-us"/>
                    </rim:Name>
                    <rim:Description/>
                    <VersionInfo versionName="1"/>
1800           </rim:Classification>
                <rim:Classification classificationScheme="urn:uuid:cccf5598-8b07-
        4b77-a05e-ae952c785ead"
                    classifiedObject="urn:uuid:08a15a6f-5b4a-42de-8f95-
        89474f83abdf"
1805               id="urn:uuid:fb7677c5-c42f-485d-9010-dce0f3cd4ad5"
                    lid="urn:uuid:fb7677c5-c42f-485d-9010-dce0f3cd4ad5"
        nodeRepresentation="Cardiology"
                    objectType="Urn:oasis:names:tc:ebxml-
        regrep:ObjectType:RegistryObject:Classification">
1810               <rim:Slot name="codingScheme">
                        <rim:ValueList>
                            <rim:Value>Connect-a-thon
        practiceSettingCodes</rim:Value>
                        </rim:ValueList>
1815               </rim:Slot>
                    <rim:Name>
                        <rim:LocalizedString charset="UTF-8" value="Cardiology"
        xml:lang="en-us"/>
                    </rim:Name>
1820               <rim:Description/>
                    <VersionInfo versionName="1"/>
               </rim:Classification>
                <rim:Classification classificationScheme="urn:uuid:f0306f51-975f-
        434e-a61c-c59651d33983"
1825               classifiedObject="urn:uuid:08a15a6f-5b4a-42de-8f95-
        89474f83abdf"
                    id="urn:uuid:0a8a8ed9-8be5-4a63-9b68-a511adee8ed5"
                    lid="urn:uuid:0a8a8ed9-8be5-4a63-9b68-a511adee8ed5"
        nodeRepresentation="34098-4"
```

```
1830                    objectType="urn:oasis:names:tc:ebxml-
      regrep:ObjectType:RegistryObject:Classification">
                     <rim:Slot name="codingScheme">
                         <rim:ValueList>
                             <rim:Value>LOINC</rim:Value>
1835                         </rim:ValueList>
                     </rim:Slot>
                     <rim:Name>
                         <rim:LocalizedString charset="UTF-8" value="Conference
      Evaluation Note"
1840                         xml:lang="en-us"/>
                     </rim:Name>
                     <rim:Description/>
                     <VersionInfo versionName="1"/>
                 </rim:Classification>
1845             <rim:ExternalIdentifier id="urn:uuid:db9f4438-ffff-435f-9d34-
      d76190728637"
                     lid="urn:uuid:db9f4438-ffff-435f-9d34-d76190728637"
                     registryObject="urn:uuid:08a15a6f-5b4a-42de-8f95-
      89474f83abdf"
1850             identificationScheme="urn:uuid:58a6f841-87b3-4a3e-92fd-
      a8ffeff98427"
                     objectType="ExternalIdentifier"
                     value="st3498702^^^&amp;1.3.6.1.4.1.21367.2005.3.7&amp;ISO">
                     <rim:Name>
1855                 <rim:LocalizedString charset="UTF-8"
      value="XDSDocumentEntry.patientId"
                             xml:lang="en-us"/>
                     </rim:Name>
                     <rim:Description/>
1860             <VersionInfo versionName="1"/>
                 </rim:ExternalIdentifier>
                 <rim:ExternalIdentifier id="urn:uuid:c3fcbf0e-9765-4f5b-abaa-
      b37ac8ff05a5"
                     lid="urn:uuid:c3fcbf0e-9765-4f5b-abaa-b37ac8ff05a5"
1865             registryObject="urn:uuid:08a15a6f-5b4a-42de-8f95-
      89474f83abdf"
                     identificationScheme="urn:uuid:2e82c1f6-a085-4c72-9da3-
      8640a32e42ab"
                     objectType="ExternalIdentifier"
1870  value="1.3.6.1.4.1.21367.2005.3.99.1.1010">
                     <rim:Name>
                         <rim:LocalizedString charset="UTF-8"
      value="XDSDocumentEntry.uniqueId"
                             xml:lang="en-us"/>
```

```
1875              </rim:Name>
                  <rim:Description/>
                  <VersionInfo versionName="1"/>
              </rim:ExternalIdentifier>
          </rim:ExtrinsicObject>
1880      </rim:RegistryObjectList>
      </AdhocQueryResponse>
```

Update to Vol 2b: 3.43.4.2.3 (Retrieve Document Set transaction)

1885 **3.43.4.2.3 Expected Actions**

A Document Repository shall retrieve the document(s) indicated in the request.

The Document Repository shall return the document or an error code in case the document could not be retrieved. The conditions of failure and possible error messages are given in the ebRS standard and detailed in ITI TF-3: 4.2.4 Error Reporting.

1890 **A document may not be available for retrieval because it is currently offline. See ITI TF-3: Table 4.2.3.2-1 (documentAvailability) for details. A Document Repository is not required to indicate in its error response the reason the document cannot be retrieved.**

Volume 3 Cross-Transaction and Content Specifications

1895

4.1 Abstract Metadata Model

Update Table 4.1.3.2-1: DocumentEntry Metadata Attribute Definition to add the following rows in alphabetical order in the table

DocumentEntry Metadata Attribute	Description	Patient identity	Provenance	Security &Privacy	Descriptive	Object Lifecycle	Exchange
documentAvailability	The status of the Document in the Document Repository					X	X
logicalID	A globally unique identifier used to identify the logical entry.		X			X	X
version	Version number of a DocumentEntry.		X			X	X

1900

Update Table 4.1.3.4-1: Folder Metadata Attribute Definition to add the following rows in alphabetical order in the table

DocumentEntry Metadata Attribute	Description	Patient identity	Provenance	Security &Privacy	Descriptive	Object Lifecycle	Exchange
logicalID	A globally unique identifier used to identify the logical entry.		X			X	X
version	Version number of a Folder.		X			X	X

1905

Add new Section 4.1.4

4.1.4 Metadata Object Versioning Semantics

One part of metadata updating is the management of metadata object versioning as specified in ebRIM 3.0. ebRIM 3.0 version control introduces the following concepts to support versioning:

1910 **Metadata Object Instance** – a single metadata object representing a single version of an object

Logical Metadata Object – the collection of metadata object instances that are the versions of a single object. Each instance is a different version.

Before the introduction of metadata update, a logical metadata object was always represented by a single instance so differentiating logical and instance was not important.

1915 A logical DocumentEntry represents a document in a repository. The logical Document Entry encompasses all the versions (Document Entry instances) that have historically represented the repository document.

An association, through its sourceObject and targetObject attributes, references metadata object instances (particular versions of the objects).

1920 Metadata versions are identified/managed through the use of two metadata attributes: logicalID and version:

logicalID

- Each object instance is assigned a logicalID along with its entryUUID (id in ebRIM terminology)

1925 - The first version of an object has entryUUID equal to logicalID

- Each metadata object instance has a unique value for the entryUUID attribute

- Each logical object has a unique value for the logicalID attribute

- Each logical object is represented by one or more object instances. So, there can be multiple object instances with the same logicalID.

1930 - All objects with the same logicalID shall be of the same type. A logicalID shall identify a group of DocumentEntry objects (a logical DocumentEntry) with the same objectType attribute or a group of Folders (a logical Folder).

- The rules for interpreting logicalID are:

  1. All object instances with the same logicalID are versions of the same logical object

1935 2. Each object instance has a unique entryUUID

  3. The first version of a logical object has logicalID = entryUUID

  4. The second and later versions of a logical object have logicalID != entryUUID

  5. If an object instance is submitted with no logicalID attribute the value for logicalID defaults to the value of the entryUUID for that object instance (becomes the first
1940   version)

---

version

- Instances of a metadata object are assigned a version through the version attribute as described in ITI TF-3: Tables 4.2.3.2-1 and 4.2.3.4-1.

1945
- The highest numbered version of an object instance shall have availabilityStatus of Approved or Deprecated All older versions shall have availabilityStatus of Deprecated.

- When updates are submitted, they reference the version being updated

- Changes shall only be accepted for the most recent version.

1950 Folder membership propagates to newer versions of a DocumentEntry. The Document Registry or Document Recipient is responsible for the propagation. This responsibility is called Association Propagation. This behavior can be overridden in the update request. See ITI TF-2b: 3.57.4.1.3.4 "Patient ID Reconciliation" for an example.

Document relationships like Addendum and Transformation propagate to newer versions of a DocumentEntry. The Document Registry or Document Recipient may be responsible for the 1955 propagation (as defined by Association Propagation). So if a DocumentEntry is linked to another as an Addendum and the metadata is updated then Document Registry or Document Recipient is responsible to make the new version of the DocumentEntry object an Addendum to the same original document. This behavior can be overridden. See Patient ID Reconciliation for an example.

1960 Changes in the availabilityStatus attribute of an object instance do not require the submission of a new instance. Changing any other attribute requires a new object instance (version) be created.

The uniqueID and logicalID attributes of a logical object (and objectType attribute of a DocumentEntry) shall not be altered through versioning. They are required to be consistent across all object instances within a logical object.

1965 **4.2.2 Association Types**

Update Table 4.2.2-1 to add the new associations as shown. (Assumes CP which updates the first column is integrated)

1970 **Table 4.2.2-1: Association Types**

Meaning	AssociationType
Membership in a Registry Package (SubmissionSet or Folder)	urn:oasis:names:tc:ebxml-regrep:AssociationType:HasMember
Replacement	urn:ihe:iti:2007:AssociationType:RPLC

Meaning	AssociationType
Transformation	urn:ihe:iti:2007:AssociationType:XFRM
Addendum	urn:ihe:iti:2007:AssociationType:APND
Replacement with Transformation	urn:ihe:iti:2007:AssociationType:XFRM_RPLC
Digital Signature	urn:ihe:iti:2007:AssociationType:signs
availabilityStatus update	**urn:ihe:iti:2010:AssociationType:UpdateAvailabilityStatus**
Submit arbitrary Association	**urn:ihe:iti:2010:AssociationType:SubmitAssociation**

4.2.3 Metadata Attributes

4.2.3.2 Document Metadata Attribute Definition

Add the following rows to ITI TF-3: Table 4.2.3.2-1 Document Metadata Attribute Definition

DocumentEntry Metadata Attribute	Description	Data Type (Table 4.2.3.1.7-2)	Coding (Section 4.2.3.1.8)	Detail
…				
creationTime	The time the author created the document. Shall have a single value.	DTM	ebRIM Slot	4.2.3.2.6
documentAvailability	**The status of the Document in the Document Repository**	**Predefined URN**	**ebRIM Slot**	**4.2.3.2.30**
entryUUID	A globally unique identifier used to identify the entry.	UUID	XML attribute	4.2.3.2.7
limitedMetadata	Indicates whether the Document Entry was created using the less rigorous requirements of metadata as defined for the Metadata-Limited Document Source.	4.2.3.2.29	ebRIM Classification	4.2.3.2.29
logicalID	**A globally unique identifier used to identify the logical entry.**	**UUID**	**XML attribute**	**4.2.3.2.31**
mimeType	MIME type of the document.	MIME type	XML attribute	4.2.3.2.15
…				
URI	The URI for the document.	URI	ebRIM Slot	4.2.3.2.27
version	**Version number of a DocumentEntry.**	**Integer**	**ebRIM VersionInfo**	**4.2.3.2.32**

1975

Add the following new sections to Section 4.2.3.2

4.2.3.2.30 DocumentEntry.documentAvailability

Description:

The status of the Document in the Document Repository. Online indicates the Document in the
1980 Document Repository is available to be retrieved. Offline indicates the Document in the
Document Repository is not available to be retrieved.

Coding:

This slot, if present, shall have a single value.

Takes one of two values: urn:ihe:iti:2010:DocumentAvailability:Online or
1985 urn:ihe:iti:2010:DocumentAvailability:Offline. If the attribute is not present in metadata its value
defaults to Online.

The following example indicates the document associated with the DocumentEntry is not
available to be retrieved.

```
1990  <rim:Slot name="documentAvailability">
        <rim:ValueList>
          <rim:Value>urn:ihe:iti:2010:DocumentAvailability:Offline</rim:Value>
        </rim:ValueList>
      </rim:Slot>
```

1995

4.2.3.2.31 DocumentEntry.logicalID

Description:

A globally unique identifier used to identify the logical entry. All versions of a DocumentEntry
carry the same logicalID but different and unique entryUUIDs. If not present in a submission,
2000 logicalID defaults to the value of the entryUUID attribute. The first version of a DocumentEntry
has logicalID equal to entryUUID. Other versions have logicalID not equal to entryUUID (also
see TF-3: 4.1.4).

Coding:

Max length is unbounded. The format of the logicalID value is UUID. LogicalID shall never be
2005 submitted in symbolic form.

The value of the logicalID is coded in the lid XML attribute on the ExtrinsicObject representing
the DocumentEntry. In the example below, the entryUUID is urn:uuid:a6e06ca8-0c75-4064-
9e5c-88b9045a96f6

The initial version of a DocumentEntry could have no LogicalID present:
```
2010  <rim:ExtrinsicObject
          id="urn:uuid:3cce0135-cedb-4a26-ba00-8698ee8dde04">
      …
      </rim:ExtrinsicObject>
```

Or could have logicalID equal to entryUUID as shown:

2015
```
<rim:ExtrinsicObject
    id="urn:uuid:3cce0135-cedb-4a26-ba00-8698ee8dde04"
    lid="urn:uuid:3cce0135-cedb-4a26-ba00-8698ee8dde04">…
</rim:ExtrinsicObject>
```

2020　One of these two forms shall be submitted in the Registry Document Set transaction.

The following form, with entryUUID (id) different from logicalID (lid), shall only be submitted in the Update Document Set transaction.

2025
```
<rim:ExtrinsicObject
    id="urn:uuid:3cce0135-cedb-4a26-ba00-8698ee8dde04"
    lid="urn:uuid:e0985823-dc50-45a5-a6c8-a11a829893bd">
…
</rim:ExtrinsicObject>
```

4.2.3.2.32 DocumentEntry.version

2030　**Description:**

Version number of a DocumentEntry. This value is assigned by the Document Registry and shall be ignored if present in a submission. The first version of a DocumentEntry shall have a value of 1. Subsequent versions get values of 2, 3, etc. This attribute shall be returned in query responses.

Coding:

2035　Max length is unbounded.

The value of the version attribute is coded in XML as the "versionName" attribute of the VersionInfo element.

```
<VersionInfo versionName="2"/>
```

2040

4.2.3.4　Folder Metadata Attribute Definition

Add the following rows to ITI TF-3: Table 4.2.3.4-1 Folder Metadata Attribute Definition

Folder Metadata Attribute	Description	Form	Coding	Detail
…				
limitedMetadata	A flag that the associated Folder was created using the less rigorous metadata requirements as defined for the Metadata-Limited	4.2.3.4.10	ebRIM Classification	4.2.3.4.10
logicalID	**A globally unique identifier used to identify the logical entry.**	**UUID**	**XML attribute**	**4.2.3.4.11**
patientId	The patientId represents the subject of care of the	CX	ebRIM ExternalIdentifi	4.2.3.1.3

Folder Metadata Attribute	Description	Form	Coding	Detail
	DocumentEntries within the Folder.		er	
…				
uniqueId	Globally unique identifier for the Folder.	OID	ebRIM ExternalIdentifier	4.2.3.4.9
version	**Version number of a Folder.**	**Integer**	**ebRIM VersionInfo**	**4.2.3.4.12**

2045 | *Add the following new sections to Section 4.2.3.4*

4.2.3.4.11 Folder.logicalID

Description:

A globally unique identifier used to identify the logical Folder. All versions of a Folder carry the same logicalID but different and unique entryUUIDs. If not present in a submission, logicalID
2050 defaults to the value of the entryUUID attribute. The first version of a Folder has logicalID (lid) equal to entryUUID (id). Other versions have logicalID not equal to entryUUID. LogicalID shall never be submitted in symbolic form (also see Section 4.1.4).

Coding:

Max length is unbounded. The format of the logicalID value is UUID. LogicalID shall never be
2055 submitted in symbolic form.

The value of the locicalID is coded in the lid XML attribute on the RegistryPackage representing the Folder.

The initial version of a Folder could have no LogicalID present:
```
<rim:RegistryPackage
2060     id="urn:uuid:3cce0135-cedb-4a26-ba00-8698ee8dde04">
…
</rim:RegistryPackage>
```

Or could have logicalID equal to entryUUID as shown:

```
2065  <rim:RegistryPackage
     id="urn:uuid:3cce0135-cedb-4a26-ba00-8698ee8dde04">
     lid="urn:uuid:3cce0135-cedb-4a26-ba00-
8698ee8dde04">
```

One of these two forms shall be submitted in the Registry Document Set transaction.

2070 The following form, with entryUUID (id) different from logicalID (lid), shall only be submitted in the Update Document Set transaction.

```
<rim:RegistryPackage
     id="urn:uuid:3cce0135-cedb-4a26-ba00-8698ee8dde04"
```

2075
```
        lid="urn:uuid:e0985823-dc50-45a5-a6c8-a11a829893bd">
…
</rim:RegistryPackage>
```

4.2.3.4.12 Folder.version

2080 **Description:**

Version number of a Folder. This value is assigned by the Document Registry and shall be ignored if present in a submission. The first version of a Folder shall have a value of 1. Subsequent versions get values of 2, 3, etc. This attribute shall be returned in query responses.

Coding:

2085 Max length is unbounded.

The value of the version attribute is coded in XML as the "versionName" attribute of the VersionInfo element.

```
<VersionInfo versionName="2"/>
```

2090

4.2.4 Error Reporting

4.2.4.1 RegistryErrors Element

ITI TF-3: Table 4.2.4.1-2: Update the Note at the bottom of the table to remove reference to XDS Metadata Update. If this causes the note to not reference any supplement, delete the note.

2095

The following additions are made to ITI TF-3: Table 4.2.4.1-2 Error Codes. Note the footnote is new too.

Error Code	Discussion	Transaction (See Note 1)
XDSMetadataUpdateError[1]	**General metadata update error. Use only when more specific error code is not available or appropriate.**	**U**
XDSPatientIDReconciliationError[1]	**Update encountered error where Patient IDs did not match**	**U**

XDSMetadataUpdateOperationError[1]	**Document Registry/ Recipient cannot decode the requested metadata update.**	**U**
XDSMetadataVersionError[1]	**The version number included in the update request did not match the existing object. One cause of this is multiple simultaneous update attempts.**	**U**
UnresolvedReferenceException[1]	**An entryUUID passed in the Delete Document Set transaction does not exist in the recipient system.**	**D**
ReferencesExistException[1]	**An entryUUID passed in the Delete Document Set transaction is referenced by an Association sourceObject or targetObject attribute.**	**D**

[1] When reporting this error, the codeContext attribute of the RegistryError element shall
2100 **contain the id attribute of the metadata object causing the error.**

Note 1:

P = Provide and Register-b

R = Register-b

2105 SQ = Stored Query

RS = Retrieve Document Set

XGQ = Cross Gateway Query

XGR = Cross Gateway Retrieve

U = Update Document Set

2110 **D = Delete Document Set**

4.3 Additional Document Sharing Requirements

Update Table 4.3.1.1-3 to add the new attributes in alphabetical order.

Table 4.3.1.1-3: Metadata Attribute Optionality

Metadata Element	Metadata Attribute	XDS DS	XDS DR	XDM MC	XDR DS	XDR MS
DocumentEntry	**documentAvailability**	**O**	**O**	**O**	**O**	**O**
DocumentEntry	**logicalID**	**O**	**O**	**O**	**O**	**O**
DocumentEntry	**version**	**O**	**O**	**O**	**O**	**O**
Folder	**logicalID**	**O**	**O**	**O**	**O**	**O**
Folder	**version**	**O**	**O**	**O**	**O**	**O**

2115

Update Table 4.3.2.1-3 to add the new attributes in alphabetical order.

Table 4.3.2.1-3: Metadata Attribute Optionality

Metadata Element	Metadata Attribute	XDS DR	XDS IG	XCA RG	MPQ DR
DocumentEntry	**documentAvailability**	**O**	**O**	**O**	**O**
DocumentEntry	**logicalID**	**O**	**O**	**O**	**O**
DocumentEntry	**version**	**O**	**O**	**O**	**O**
Folder	**logicalID**	**O**	**O**	**O**	**O**
Folder	**version**	**O**	**O**	**O**	**O**

2120 *Add new Section 4.3.3 within 4.3 to explain requirements on the Update Document Set transaction*

4.3.3 Requirements on Update Document Set Transaction

4.3.3.1 SubmitAssociation

When an Association is submitted via the Update Document Set transaction Submit Associations
2125 operation (see ITI TF-2b: 3.57.4.1.3.3.6) to repair metadata it is linked to the SubmissionSet object via the SubmitAssociation Association type.

Example of SubmitAssociation association type

```
<Association
  id="ID_01"
  associationType="urn:ihe:iti:2010:AssociationType:SubmitAssociation"
  sourceObject="SubmissionSet"
  targetObject="apnd_assoc"/>
```

2130

2135
```
<Association
  id="apnd_assoc"
  associationType="urn:ihe:iti:2007:AssociationType:APND"
  sourceObject="urn:uuid:8ace0e1c-7dd9-47a0-9e7d-d673beb052e0"
  targetObject="urn:uuid:aa0da13b-51b0-4c2e-868c-cef8d7e1bc3d"/>
```

2140

The Document Registry shall verify upon submission:

- The SubmitAssociation association sourceObject attribute references the SubmissionSet object

2145
- The SubmitAssociation association targetObject attribute references an association present in the submission

- The referenced association shall have:

  - SourceObject and targetObject attributes reference objects already present in the recipient system (which implies UUID format).

  - The associationType is appropriate for the objects referenced

2150 If the SubmissionSet is submitted in an Update Document Set transaction and the update includes a DocumentEntry or Folder then the HasMember Association linking the DocumentEntry/Folder to the SubmissionSet shall include a PreviousVersion annotation (see Section 4.1.6.4.4.6 for details).

If the above example were applied to a Folder, it would not contain the SubmissionSetStatus
2155 annotation, which only applies to DocumentEntries.

4.3.3.2 Trigger Associations

Trigger Associations shall only be used in the Update Document Set transaction. They are always attached to the SubmissionSet object through their sourceObject attribute. The association type indicates the action to be triggered in the Document Registry or Document
2160 Recipient Actor. The targetObject attribute identifies the object in the receiving actor to be affected.

4.3.3.2.1 UpdateAvailabilityStatus

The object referenced by the targetObject attribute of this Association shall have its availabilityStatus attribute set to a new value. The targetObject attribute shall reference an object
2165 already present in the receiving actor. This object shall be of type Association, DocumentEntry or Folder. If it is of type DocumentEntry or Folder, it shall be the most recent version of that object.

This Association shall have two annotations: NewStatus and OriginalStatus. The receiving actor shall verify that the availabilityStatus of the targetObject matches the OriginalStatus annotation.

2170   The targetObject shall then have its availabilityStatus attribute changed to the value in the
NewStatus annotation.

Example

```
<Association
  associationType="urn:ihe:iti:2010:AssociationType:UpdateAvailabilityStatus"
  sourceObject="SubmissionSet"
  targetObject="urn:uuid:e0985823-dc50-45a5-a6c8-a11a829893bd">
    <Slot name="NewStatus">
       <ValueList>
          <Value>urn:oasis:names:tc:ebxml-
regrep:StatusType:Deprecated</Value>
       </ValueList>
    </Slot>
    <Slot name="OriginalStatus">
       <ValueList>
          <Value>urn:oasis:names:tc:ebxml-regrep:StatusType:Approved</Value>
       </ValueList>
    </Slot>
</Association>
```

2190

This example shows the deprecation of either a DocumentEntry, Folder, or Association.

Accessing the Registry Object behind the targetObject attribute UUID is the only way to know
which of these three kinds of objects is being updated.

2195   **NewStatus**

This annotation shall only be used on SubmissionSet UpdateAvailabilityStatus Associations. It
shall be coded as a Slot with a single value. It carries the new value of availabilityStatus to be
assigned to the registry object identified by the targetObject attribute.

2200   **OriginalStatus**

This annotation shall only be used on SubmissionSet UpdateAvailabilityStatus Associations. It
shall be coded as a Slot with a single value. It carries the expected current value of
availabilityStatus of the registry object identified by the targetObject attribute.

2205 **PreviousVersion**

This annotation is constructed as a Slot with name PreviousVersion containing a single value, the previous version number: 1, 2, 3, etc. This annotation is required on the SubmissionSet HasMember Association linked to DocumentEntry or Folder objects in an Update Document Set transaction. The DocumentEntry or Folder object is an update to an existing object and the
2210 PreviousVersion value documents the version number of the object being updated. The new version (present in the submission) will get the value PreviousVersion + 1.

Example

```
<Association
  associationType="urn:oasis:names:tc:ebxml-regrep:AssociationType:HasMember"
  sourceObject="SubmissionSet01"
  targetObject="Document01">
    <Slot name="SubmissionSetStatus">
        <ValueList>
            <Value>Original</Value>
        </ValueList>
    </Slot>
    <Slot name="PreviousVersion">
        <ValueList>
            <Value>1</Value>
        </ValueList>
    </Slot>
</Association>
```

The DocumentEntry with id=Document01 (not shown) is an update. The PreviousVersion is 1 so
2230 the new version will get a version number of 2.

AssociationPropagation

Example SubmissionSet – DocumentEntry HasMember Association as part of an update
2235 **including AssociationPropagation annotation**

```
<Association
  associationType="urn:oasis:names:tc:ebxml-regrep:AssociationType:HasMember"
  sourceObject="SubmissionSet01"
  targetObject="Document01">
    <Slot name="SubmissionSetStatus">
        <ValueList>
            <Value>Original</Value>
        </ValueList>
    </Slot>
```

```
2245        <Slot name="PreviousVersion">
              <ValueList>
                 <Value>1</Value>
              </ValueList>
           </Slot>
2250       <Slot name="AssociationPropagation">
              <ValueList>
                 <Value>no</Value>
              </ValueList>
           </Slot>
2255    </Association>
```

The default behavior for association propagation is defined in ITI TF-2b: 3.57.4.1.2.2. This example shows the SubmissionSet HasMember association for a new version of a DocumentEntry. The example AssociationPropagation annotation disables association
2260   propagation for this object (Document01).