Integrating the Healthcare Enterprise

5 # IHE IT Infrastructure
Technical Framework Supplement

10 # XDS Metadata Update

15 # Rev. 1.12 – Trial Implementation

20 Date: July 2, 2021
Author: ITI Technical Committee
Email: iti@ihe.net

25

Foreword

This is a supplement to the IHE IT Infrastructure Technical Framework V17.0. 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 July 2, 2021 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://www.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 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 IT Infrastructure Technical Framework can be found at
https://profiles.ihe.net/ITI/TF/index.html.

55

CONTENTS

Introduction to this Supplement

This supplement updates the XDS and XDR Profiles to add support for updating metadata. The Document Administrator is the source of the new Update Document Set transaction for requesting a metadata update to the Document Registry. These new capabilities are assigned to this new actor to enable tighter authentication and authorization control over its use.

In XDS, the Update Document Set [ITI-57] transaction is sent to the Document Registry and updates are visible through the Registry Stored Query [ITI-18] transaction. In XDR, Update Document Set is 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.

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

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

Open Issues and Questions

None

Closed Issues

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

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

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

MV031: Metadata Update will require significant authentication/authorization challenges. Should these be mandated by this profile or left to be decided upon by developers?

Recommendations made in security sections. No mandates made.

MV033: Since there is a significant amount of shared content and intertwined concepts should
200 *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.*

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.
205 *Is this correct or should the Document Administrator simply be required to support both transactions? Resolution: (03/2017) The Delete Document Set transaction [ITI-62] has been renamed and moved to the Remove Metadata and Documents Profile. As a result, this condition no longer exists. To claim support as an XDS or XDR Document Administrator, the Update Document Set transaction is the only transaction required to be supported.*

210 *MV035: Should XCA declare options for support of metadata versions in the query response? 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*
215 *now). Could rename the option in the other supplement to be something like "Enhanced 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 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 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 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. Resolution: (03/2017) The Delete Document Set transaction [ITI-62] has been renamed and moved to the Remove Metadata and Documents Profile. This open issue is being address in this profile.

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

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. Resolution: (03/2017) This CP is being resolved in the Remove Metadata and Documents Profile.

220

225

230

235

240

IHE Technical Frameworks General Introduction

The [IHE Technical Framework General Introduction](#) is shared by all of the IHE domain technical frameworks. Each technical framework volume contains links to this document where appropriate.

9 Copyright Licenses

IHE technical documents refer to, and make use of, a number of standards developed and published by several standards development organizations. Please refer to the IHE Technical Frameworks General Introduction, [Chapter 9 - Copyright Licenses](#) for copyright license information for frequently referenced base standards. Information pertaining to the use of IHE International copyrighted materials is also available there.

10 Trademark

IHE® and the IHE logo are trademarks of the Healthcare Information Management Systems Society in the United States and trademarks of IHE Europe in the European Community. Please refer to the IHE Technical Frameworks General Introduction, [Chapter 10 - Trademark](#) for information on their use.

IHE Technical Frameworks General Introduction Appendices

The IHE Technical Framework General Introduction Appendices are components shared by all of the IHE domain technical frameworks. Each technical framework volume contains links to these documents where appropriate.

> *Update the following appendices to the General Introduction as indicated below. Note that these are **not** appendices to this domain's Technical Framework (TF-1, TF-2, TF-3 or TF-4) but rather, they are appendices to the IHE Technical Frameworks General Introduction located here.*

Appendix A – Actors

> *Add the following **new or modified** actors to the IHE Technical Frameworks General Introduction Appendix A:*

Actor	Definition
Document Administrator	The Document Administrator is an actor capable of updating and/or removing metadata from the Document Registry. This actor may also be capable of removing associated documents from the Document Repository

Appendix B – Transactions

> *Add the following **new or modified** transactions to the IHE Technical Frameworks General Introduction Appendix B:*

Transaction	Definition
Update Document Set [ITI-57]	The Update Document Set transaction is used by the Document Administrator to request metadata updates.

Appendix D – Glossary

> *Add the following **new or modified** glossary terms to the IHE Technical Frameworks General Introduction Appendix D:*

No new glossary entries.

Volume 1 – Integration Profiles

1.10 History of Annual Changes

285

Add the following bullet to the end of the bullet list in Section 1.10

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

10 Cross-Enterprise Document Sharing (XDS.b)

…

290 ## 10.1 Actors/Transactions

Update the following diagram to add the Document Administrator and the new transaction.
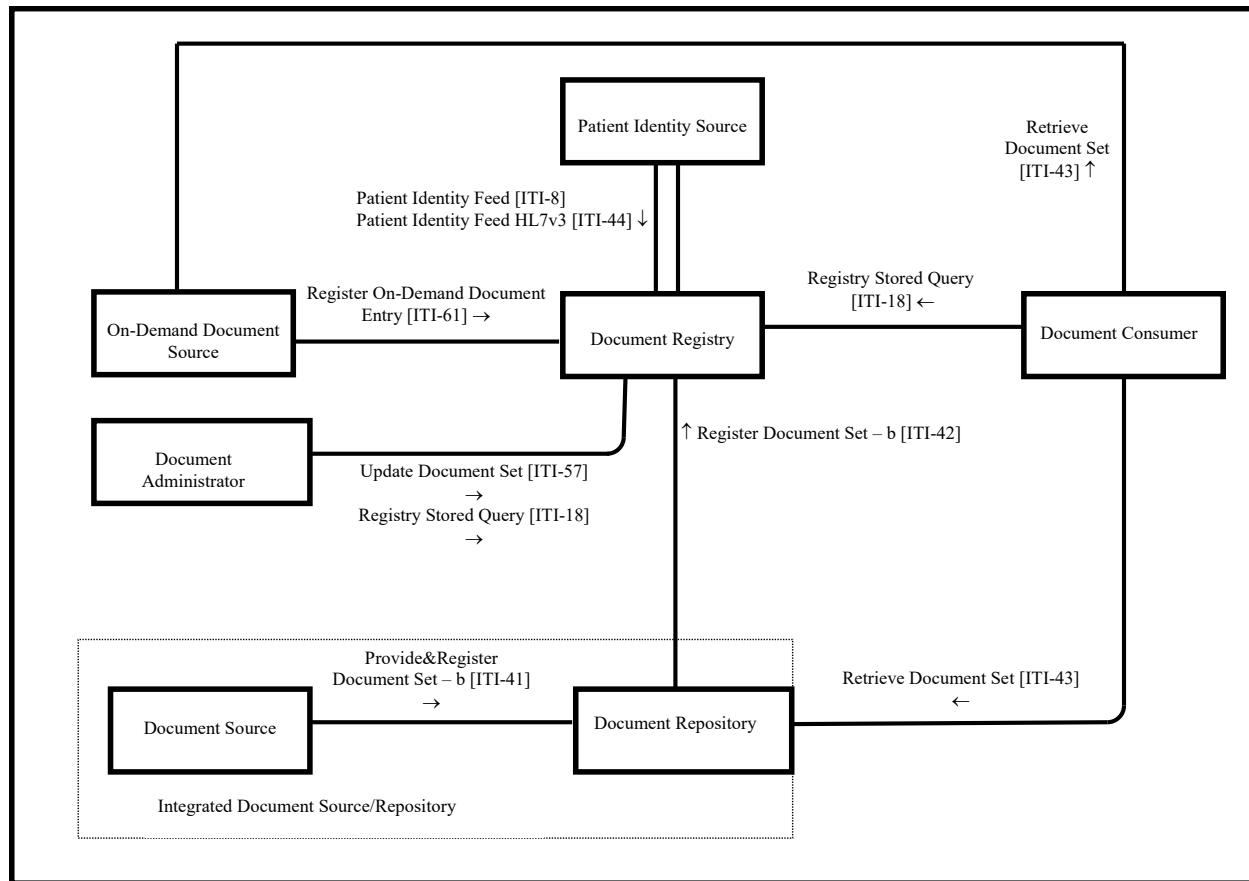
Figure 10.1-1b: Cross-Enterprise Document Sharing –b (XDS.b) Diagram

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

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

Actors	Transactions	Optionality	Section
…	\		
Document Registry	Register Document Set-b [ITI-42]	R	ITI TF-2: 3.42
	Registry Stored Query [ITI-18]	R	ITI TF-2: 3.18
	Patient Identity Feed [ITI-8]	O (Note 2)	ITI TF-2: 3.8

Actors	Transactions	Optionality	Section
	Patient Identity Feed HL7v3 [ITI-44]	O (Note 2)	ITI TF-2: 3.44
	Update Document Set [ITI-57]	**O**	**ITI TF-2: 3.57**
…			
Document Administrator	**Update Document Set [ITI-57]**	**R**	**ITI TF-2: 3.57**
	Registry Stored Query [ITI-18]	**O**	**ITI TF-2: 3.18**

…

Add Sections 10.1.1.8 and 10.1.2.9

10.1.1.8 Document Administrator

300　The Document Administrator supports metadata update by issuing the Update Document Set [ITI-57] transaction to the Document Registry and shall be capable of generating at least one of the operations documented in ITI TF-2: 3.57.4.1.3.3.

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

305　### 10.1.2.9 Update Document Set

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

This transaction may also include Associations. For example, an Association may be included to deprecate an On-Demand DocumentEntry.

310　## 10.2 XDS Integration Profile Options

Update Table 10.2-1b as follows:

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

Actor	Options	Vol. & Section
…		
Document Registry	Patient Identity Feed (Note 1)	ITI TF-2: 3.8
	Patient Identity Feed HL7v3 (Note 1)	ITI TF-2: 3.44
	Asynchronous Web Services Exchange	ITI TF-1: 10.2.5
	Reference ID	ITI TF-1: 10.2.6
	On-Demand Documents	ITI TF-1: 10.2.7
	Document Metadata Update	**ITI TF-1: 10.2.10**
…		
Document Consumer	Basic Patient Privacy Enforcement	ITI TF-1: 10.2.9
	Basic Patient Privacy Proof	ITI TF-2: 3.18.4.1.3.6

Actor	Options	Vol. & Section
	Asynchronous Web Services Exchange	ITI TF-1: 10.2.5
	On-Demand Documents	ITI TF-1: 10.2.7
	Document Metadata Update	**ITI TF-1: 10.2.10**
…		
Document Administrator	**No options defined**	**--**

Add Section 10.2.10

315 ## 10.2.10 Document Metadata Update Option

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

- Accept metadata updates via the Update Document Set [ITI-57] transaction (see ITI TF-2: 3.57.4.1.3 for details). All operations documented in ITI TF-2: 3.57.4.1.3.3 shall be supported.

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

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

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

…

10.4 General Principles

…

330 *Add Section 10.4.14*

10.4.14 Metadata Update

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

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

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

- 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
340 Association must be deprecated and a new Association installed.

- 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:

- Update patient demographics

- Update confidentiality code

- Deprecate a document without replacing it

- Deprecate the membership of a DocumentEntry from a Folder

- Deprecate a relationship (addendum, transformation, etc.)

- Update the Patient ID

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

- 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 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 deprecating 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 best performed within the Document Registry. 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 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 the updating of availabilityStatus.

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

345

350

355

360

365

370

375

---

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 available through the audit logs prescribed in the Audit Trail and Node Authentication (ATNA) Profile.

…

10.5 Implementation Strategies

…

Add Section ITI TF-1: 10.5.1

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 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.

…

395 **15 Cross-Enterprise Document Reliable Interchange (XDR)**

…

15.1 Actors/Transactions

In ITI TF-1:15.1 replace the XDR Actor Diagram in Figure 15.1-1 with the following:

…

```
┌─────────────────────┐                                                      ┌─────────────────────┐
│                     │   Provide and Register Document Set-b [ITI-41] →     │                     │
│   Document Source   │─────────────────────────────────────────────────────│                     │
│                     │                                                      │                     │
└─────────────────────┘                                                      │                     │
┌─────────────────────┐                                                      │                     │
│   Metadata-Limited  │   Provide and Register Document Set-b [ITI-41] →     │  Document Recipient │
│   Document Source   │─────────────────────────────────────────────────────│                     │
│                     │                                                      │                     │
└─────────────────────┘                                                      │                     │
┌─────────────────────┐                                                      │                     │
│     Document        │        Update Document Set  [ITI-57] →               │                     │
│   Administrator     │─────────────────────────────────────────────────────│                     │
│                     │                                                      │                     │
└─────────────────────┘                                                      └─────────────────────┘
```
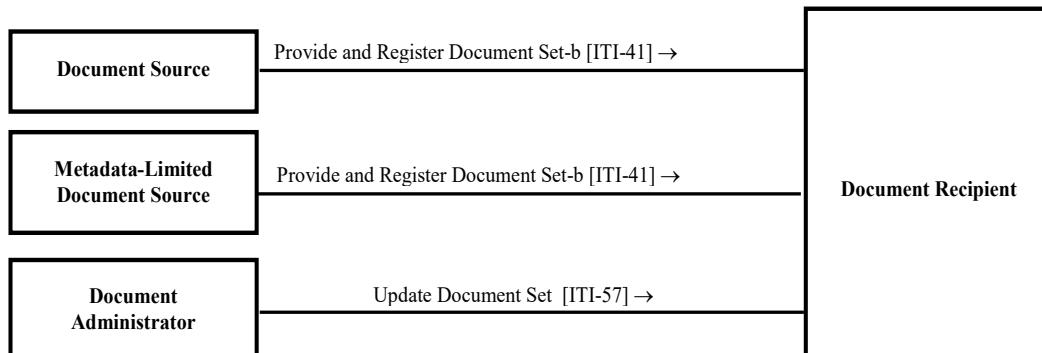
400

Update ITI TF-1: Table 15.1-1, including adding the note beneath the table.

…

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

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

405

Update ITI TF-1: Table 15.2-1:

…

Table 15.2-1: XDR - Actors and Options

Actor	Options	Volume & Section
…		
Document Recipient	Basic Patient Privacy Enforcement	ITI TF-1: 15.2.2
	Accepts Limited Metadata	ITI TF-1: 15.2.3

Actor	Options	Volume & Section
	Document Metadata Update	**ITI TF-1: 15.2.4**
Document Administrator	**No options defined**	--

410 | *In ITI TF-1, add Section 15.2.4* |

15.2.4 Document Metadata Update Option

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

The Document Recipient shall be capable of all the operations for the Update Document Set
415 [ITI-57] transaction documented in ITI TF-2: 3.57.4.1.3.3.

| *In ITI TF-1, add Sections 15.6* |

15.6 Metadata Update

The Document Metadata Update Option documents the use of the Update Document Set [ITI-57]
420 transaction when communicating with a Document Recipient. Section ITI TF-1: 10.4.14 documents general principles of this transaction 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. **T**he Document Source will have to maintain the metadata submitted to the Document Recipient to permit the proper coding of future updates. To use this transaction the
425 Document Source shall:

- 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.

- Retain version numbers for all transmitted versioned objects so the proper version
430 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.

Volume 2 – Transactions

435   *Add Section 3.57*

3.57 Update Document Set [ITI-57]

This section corresponds to transaction [ITI-57] of the ITI Technical Framework. Transaction [ITI-57] is used by the Document Administrator, Document Registry, and Document Recipient.

3.57.1 Scope

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

- A SubmissionSet object that organizes the content of the update

- Zero or more updated DocumentEntry objects

- Zero or more updated Folder objects

445   - Zero or more other Associations that trigger other updates

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

3.57.2 Use Case Roles
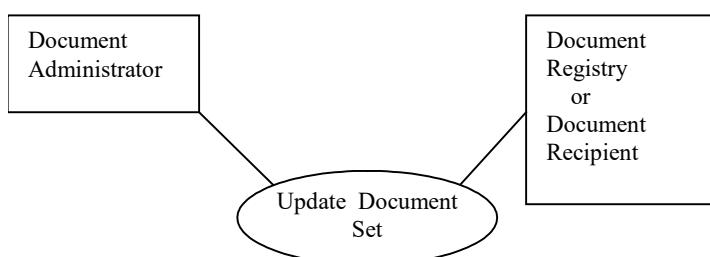
450

Actor: Document Administrator

Role: Issues metadata updates

Actor: Document Registry or Document Recipient

Role: Accepts metadata updates

455   ### 3.57.3 Referenced Standard

Implementers of this transaction shall comply with all requirements described in ITI TF-2: 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-2: 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 Messages
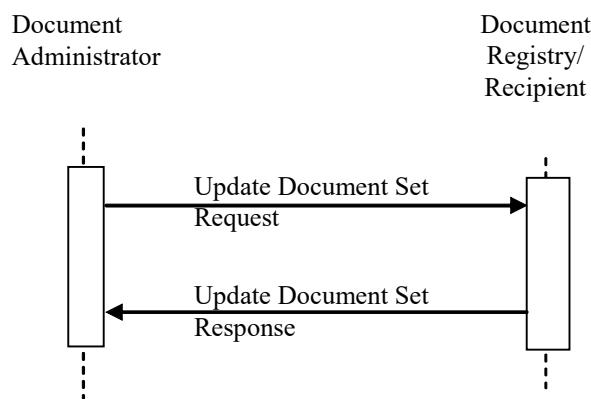
460

Figure 3.57.4-1: 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 updates to registry metadata:

465
- 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

470 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:

- Section 3.57.4.1.3.3.1 – Update DocumentEntry Metadata

- Section 3.57.4.1.3.3.2 – Update DocumentEntry availabilityStatus

475
- Section 3.57.4.1.3.3.3 – Update Folder Metadata

- Section 3.57.4.1.3.3.4 – Update Folder availabilityStatus
- Section 3.57.4.1.3.3.5 – Update Association availabilityStatus
- Section 3.57.4.1.3.3.6 – Submit Associations

3.57.4.1.2 Message Semantics

480 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: 4.1.5 for Metadata Versioning Semantics. Metadata updates are submitted as contents of a SubmissionSet.

ITI TF-3: Table 4.3.1-3 details the requirements on metadata attributes for each actor submitting
485 metadata. In that table, the XDS Document Administrator shall conform to the XDS Document Repository (XDS DR) column for Stable Documents and shall conform to the XDS On-Demand Document Source (XDS OD) column for On-Demand Documents. The XDR Document Administrator shall conform to the XDR Metadata-Limited Document Source (XDR MS) column.

490 #### 3.57.4.1.2.1 Message Definition

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>
```

Figure 3.57.4.1.2.1-1: Example Update Document Set Request message

Note:

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

510
- Each update contained in the message includes an association linked to the 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

515
- If the targetObject attribute of the association references an object contained in the update, that object also helps control the update

ITI TF-3: 4.1.5 documents the semantics of metadata update.

3.57.4.1.3 Expected Actions

520 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 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.

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

- logicalID

- version

- availabilityStatus

530 which require special handling dictated by metadata update. The receiving actor shall be capable 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, ITI TF-3: 4.1.5 shall be consulted for an understanding of the 535 semantics of metadata update.

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, 540 specializes and extends these common rules.

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.

545
3. The only way to update the metadata attributes of a DocumentEntry or Folder is by 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.

550

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 connection between two objects that is no longer current.

555

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

560

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.

565

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.

570

11. A single Update Document Set transaction may contain both an Update DocumentEntry 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.

575

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.

580

13. A single Update Document Set transaction shall not contain multiple Update (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.

585 15. At any point in time there shall be at most one version of a logical DocumentEntry object 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
590   recent version.

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.

595 **3.57.4.1.3.1.1 Rules for Update Planning**

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.

600 The phrase "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
605 Association 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

610
- 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:

615
- 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
620   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 updates.

The result is that the Document Registry or Document Recipient 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 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:

  o First installing updated versions of the objects

  o Second performing Association Propagation to interconnect the affected objects

- Reject update requests where two related updates have different values for AssociationPropagation (see Section 3.57.4.1.3.1.3.4 for details)

3.57.4.1.3.1.2 UpdateAvailabilityStatus

If a Trigger Association (see Section 3.57.4.1.3.3) triggers an update to the availabilityStatus of an object, the object referenced by the targetObject attribute shall have its availabilityStatus attribute set to a new value. The targetObject attribute shall reference an object 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. The targetObject shall then have its availabilityStatus attribute changed to the value in the NewStatus annotation.

```
<Association id="triggerAssociation"
  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">
```

```
660        <ValueList>
               <Value>urn:oasis:names:tc:ebxml-regrep:StatusType:Approved</Value>
           </ValueList>
       </Slot>
</Association>
```

665   **Figure 3.57.4.1.3.1.2-1: Deprecation Example**

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.

3.57.4.1.3.1.3 Annotations

670 The Annotations in this section are added to Associations for one of three reasons:

To specify the new state of the target object

To specify the expected prior state of the target object

To force some processing by the recipient of the submission (Document Registry or Document Recipient Actors)

675 Annotations are specific to the type of object referenced by the source and target objects and to the association type.

Annotations are coded as Slots inside the Association object.

3.57.4.1.3.1.3.1 NewStatus

This annotation shall only be used on SubmissionSet UpdateAvailabilityStatus Associations. It
680 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.

3.57.4.1.3.1.3.2 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
685 availabilityStatus of the registry object identified by the targetObject attribute.

3.57.4.1.3.1.3.3 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
690 transaction. The DocumentEntry or Folder object is an update to an existing object and the PreviousVersion value documents the version number of the object being updated. The new version (present in the submission) will get the value PreviousVersion + 1.

```
<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>
```

Figure 3.57.4.1.3.1.3.3-1: PreviousVersion Example

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

3.57.4.1.3.1.3.4 AssociationPropagation

The default behavior for association propagation is defined in Section 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 propagation for this object (Document01).

```
<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>
    <Slot name="AssociationPropagation">
       <ValueList>
```

```
        <Value>no</Value>
      </ValueList>
    </Slot>
</Association>
```

735

Figure 3.57.4.1.3.1.3.4-1: Example SubmissionSet – DocumentEntry HasMember Association as part of an update including AssociationPropagation annotation

740 **3.57.4.1.3.2 Error Reporting**

Any errors occurring in the processing of an Update Document Set Request shall cause the entire 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

745 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 Request. When the Document Registry or Document Recipient receives a submission and cannot decode the operation following the rules stated below, it shall return the error code XDSMetadataUpdateOperationError.

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

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

- Section 3.57.4.1.3.3.2 - Update DocumentEntry Status

- Section 3.57.4.1.3.3.3 - Update Folder Metadata

- Section 3.57.4.1.3.3.4 - Update Folder Status

755 - Section 3.57.4.1.3.3.5 - Update Association Status

- Section 3.57.4.1.3.3.6 - Submit new Association object(s)

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
760 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 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.

765 Trigger Associations "trigger" a side effect in the receiving actor. Trigger Associations shall only be used in the Update Document Set transaction. The sourceObject attribute of the Trigger Association shall contain the id of the SubmissionSet object. The association type indicates the action to be triggered in the Document Registry or Document Recipient. The targetObject attribute identifies the object in the receiving actor to be affected.

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

Actions - the actions taken by the Document Registry or Document Recipient if the Trigger and Preconditions are met.

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

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
780 creation of a new version.

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

3.57.4.1.3.3.1 Update DocumentEntry Metadata

3.57.4.1.3.3.1.1 Semantics

785 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) 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
790 Metadata operation in a submission:

- Submission contains a DocumentEntry object.

  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
795   PreviousVersion. This Slot has a single value, the version number of the previous version, the one being replaced.

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. Note that the validation rules may result in multiple error conditions. The Document
800 Administrator should not assume that error(s) returned are the only error conditions that exist on the receiving actor. Unless a more specific code is provided within the validation, the receiving actor shall return the error code, *XDSMetadataUpdateError*, for any error returned during processing.

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

Copyright © 2021: IHE International, Inc.

2. Submitted DocumentEntry and existing DocumentEntry have the same values for the 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
810   XDSMetadataVersionError.

4. The Document Registry or Document Recipient shall verify that the new DocumentEntry 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
815 availabilityStatus attribute is set to the value found in the existing DocumentEntry.

3.57.4.1.3.3.1.5 Association Propagation

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
820 Document Administrator takes full responsibility for linking the updated DocumentEntry object to existing metadata.

The receiving actor scans for non-deprecated HasMember Associations linking the existing DocumentEntry to a Folder. When found, a new FD-DE HasMember Association is generated linking the included DocumentEntry to the same Folder. In addition, a new SS-HM HasMember
825 Association is generated linking the generated FD-DE HasMember Association to the current Submission Set.

The receiving actor scans for non-deprecated relationship associations (see ITI TF-3: 4.2.2.2) linked to the existing DocumentEntry. When found, these associations are replicated referencing the submitted DocumentEntry instead of the existing DocumentEntry. This causes the submitted
830 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

AvailabilityStatus of the existing DocumentEntry is changed to Deprecated.

3.57.4.1.3.3.1.7 Patient ID Management

835 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 the XDSPatientIDReconciliationError error code.

3.57.4.1.3.3.2 Update DocumentEntry AvailabilityStatus

840 **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.

3.57.4.1.3.3.2.2 Trigger

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

- Submission contains an 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,
850 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.

  3. The Association contains a Slot with name NewStatus containing a single value, the namespace qualified availabilityStatus value to be assigned to the DocumentEntry
855 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

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
860 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

865 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
870 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.

875 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
880 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
885 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
890 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.

3.57.4.1.3.3.3.3 Preconditions

895 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.

    3. The value of the PreviousVersion Slot matches the version number of the existing Folder.
900 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 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 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 FD-DE HasMember Association is generated linking the included Folder to the DocumentEntry. In addition, a new SS-HM HasMember Association is generated linking the generated FD-DE HasMember Association to the current Submission Set.

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

The 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 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

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:

- Submission contains an 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. 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.

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

3.57.4.1.3.3.4.3 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 Folder in the registry with availabilityStatus matching the value of OriginalStatus.

945   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

950   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.

955   ### 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.

960   ### 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.

965   ### 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
970   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.

975

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:

980

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

2. One of these two conditions shall be true:

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

   o The existing Association is a relationship type Association

985

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

990 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
995 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
1000 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:

1005
- 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.

1010

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

<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"/>
```

Figure 3.57.4.1.3.3.6.1-1: SubmitAssociation Association Example

3.57.4.1.3.3.6.2 Trigger

The following rules shall be used by the receiving actor to detect a Submit Associations
1025 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

1030 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) and the association type is appropriate for the objects referenced.

2. The sourceObject and targetObject shall not be deprecated and the association type is
1035 appropriate for the objects referenced.

3. The sourceObject and targetObject shall not reference the current SubmissionSet object.

3.57.4.1.3.3.6.4 Actions

Save new association.

1040 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 3.57.4.1.3.1.3.3 for details).

3.57.4.1.3.3.6.5 Association Propagation

None

1045 ### 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 1050 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.

1055 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 1060 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:

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

1065 Rule #1 above is refined:

1. 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.

1070   2. The Update Objects Request transaction is an atomic operation:

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

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

        o   If any part of the transaction fails, it shall leave the registry in the state it was at the
1075   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
1080 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
1085 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.

1090 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
1095 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
1100   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.

1105
- Create a new Folder with a new Patient ID to hold part of existing Folder contents that get a new Patient ID

- Remove DocumentEntry membership from a Folder by deprecating the HasMember association

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

1110 **3.57.4.1.3.4.1 Unconnected DocumentEntry Example**

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
1115 Rule #4 in Section 3.57.4.1.3.4.

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
1120 original DocumentEntry and the amendment DocumentEntry (with new Patient ID) and a new 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

1125 Updating the Patient ID on a DocumentEntry that has an APND relationship with another 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:

1130
- Deprecate the APND association and update the original DocumentEntry with a new Patient ID. The amendment DocumentEntry loses its status as an amendment but 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.

1135
- 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 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.

3.57.4.2 Update Document Set Response

3.5.7.4.2.1 Trigger Events

1140 After receiving the Update Document Set request, the receiving actor processes the update capturing any errors that occur. When processing is complete, the response message is triggered.

3.5.7.4.2.2 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
1145 used. Errors reported shall include the entryUUID attribute value of the object triggering the error reported in the location attribute of the RegistryError element.

3.5.7.4.2.3 Expected Actions

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

3.57.5 Protocol Requirements

1150 The Update Document Set request and response will be transmitted using Synchronous Web Service Exchange, according to the requirements specified in ITI TF-2: 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
1155   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
1160   Document Set Response message shall be defined as "urn:ihe:iti:2010:UpdateDocumentSetResponse"

- The /definitions/binding/operation/soap12:operation/@soapActionRequired attribute shall be defined as "false"

1165 **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>
```

1180 **Figure 3.57.5.1-1: Update Document Set Response – Example SOAP message**

3.57.5.2 Message Examples

A collection of examples is available online. See ITI TF-2: Appendix W.

3.57.6 Actor Requirements

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

1185 **3.57.6.1 Document Administrator**

A Document Administrator 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 supports the Document Metadata Update Option shall be capable of
1190 all of the operations documented in Section 3.57.4.1.3.3.

3.57.6.3 Document Recipient

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

3.57.7 Security Considerations

1195 Relevant XDS Affinity Domain security considerations are discussed in the XDS Security Considerations Section (see ITI TF-1: 10.7).

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
1200 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 using the Provide and Register Document Set-b [ITI-41] 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 Cross-Enterprise User Authorization (XUA) Profile by policy in most environments.

Some pertinent risks are:

- 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

3.57.7.1 Audit Record Considerations

The Update Document Set transaction is PHI-Export event, as defined in ITI TF-2: Table 3.20.4.1.1.1-1 with the following exceptions.

3.57.7.1.1 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:

	UserID	U	not specialized
Source **AuditMessage/** **ActiveParticipant**	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

	UserID	M	Identity of the human that initiated the transaction
Human **Requestor** **(if known)** **AuditMessage/** **ActiveParticipant**	*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
	NetworkAccessPointTypeCodes	*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")
	NetworkAccessPointTypeCodes	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*

1220

SubmissionSet **(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*

3.57.7.1.2 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)			

Where:

Source **AuditMessage/** **ActiveParticipant**	*UserID*	*U*	*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

1225

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*

SubmissionSet **AuditMessage/Participan tObjectIdentification**	ParticipantObjectTypeCode	M	"2" (System)
	ParticipantObjectTypeCodeRol e	M	"20" (job)
	ParticipantObjectDataLifeCycl e	*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*

3.18 Registry Stored Query [ITI-18]

…

1230   *Update ITI Vol 2: 3.18.2 Use Case Roles*

3.18.2 Use Case Roles

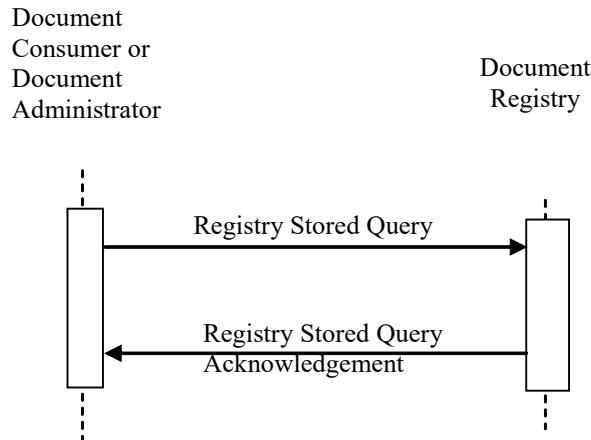**Actor**: Document Consumer **or Document Administrator**

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
1235   objects.

Actor: Document Registry

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

Update ITI Vol 2: 3.18.4 to add Document Administrator

1240 **3.18.4 Messages**

Update to ITI Vol 2: 3.18.4.1.1

1245 **3.18.4.1.1 Trigger Events**

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

…

1250 *Add new Section 3.18.4.1.2.3.5.1 and subsections in ITI TF-2 after existing Section 3.18.4.1.2.3.5 "Coding of Single/Multiple Values" 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

1255 **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 Registry Stored Query [ITI-18] transaction so that interoperability is not degraded by the installation of more modern Document Registry implementations alongside older Document Consumer implementations.

1260 The introduction of metadata update functionality is an incompatible update to the Document Sharing 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
  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
  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.

- Versioning of Folder objects - until the introduction of the optional Update Document Set
  [ITI-57] transaction, new versions of Folder objects could not be created.

- Versioning of DocumentEntry objects – until the introduction of the optional Update
  Document Set [ITI-57] transaction, DocumentEntry objects could be deprecated by
  replacement but not by versioning.

These capabilities introduce metadata patterns that a pre-metadata update Document 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.

The default value of $MetadataLevel is 1 (one).

 If $MetadataLevel is 1 (one), the Document Registry shall not return the following metadata
patterns in the Stored Query response:

- DocumentEntry and Folder objects that are not the latest version

- DocumentEntry objects that have documentAvailability of
  `urn:ihe:iti:2010:DocumentAvailability:Offline`

- Folder objects that have availabilityStatus of `urn:oasis:names:tc:ebxml-regrep:StatusType:Deprecated`

- Association objects that have availabilityStatus of `urn:oasis:names:tc:ebxml-regrep:StatusType:Deprecated`

Note that these requirements only prevent the Document Registry from returning objects that are
meaningless to, or inappropriate for, a Document Consumer that only supports
$MetadataLevel=1. Because the Document Registry does not return these objects, however, the
Document Consumer may not receive objects that it expects to receive.

For example:

1300
- The Document Consumer may receive a FD-DE HasMember Association in the response to a GetAssociations query but not receive the referenced Folder in the response to a subsequent GetFolders query because the Folder has availabilityStatus of Deprecated.

- The Document Consumer may receive a Relationship Association in the response to a GetAssociations query but not receive the referenced DocumentEntry in the response to a
1305   subsequent GetDocuments query because the DocumentEntry has documentAvailability of Offline.

- The Document Consumer may receive a deprecated DocumentEntry in the response to a GetDocumentsAndAssociations query but not receive any RPLC or XFRM_RPLC Associations because the DocumentEntry was deprecated by an Update DocumentEntry
1310   Status operation rather than by the submission of a replacement DocumentEntry.

Individual Stored Queries may define more specific behavior if appropriate.

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.

1315 **3.18.4.1.2.3.5.1.2 Stored Query parameters added by Document Metadata Update Option**

The following parameters are added by the Document Metadata Update Option.

Stored Query	Added parameters
FindDocuments	$XDSDocumentEntryDocumentAvailability
	$MetadataLevel
FindFolders	$MetadataLevel
GetAll	$XDSAssociationStatus
	$MetadataLevel
GetDocuments	$XDSDocumentEntryLogicalID
	$MetadataLevel
GetFolders	$XDSFolderLogicalID
	$MetadataLevel
GetAssociations	$XDSAssociationStatus
	$MetadataLevel
GetDocumentsAndAssociations	$XDSAssociationStatus
	$MetadataLevel

Stored Query	Added parameters
GetSubmissionSets	$MetadataLevel
GetSubmissionSetAndContents	$MetadataLevel
GetFolderAndContents	$XDSAssociationStatus
	$MetadataLevel
GetFoldersForDocument	$XDSAssociationStatus
	$MetadataLevel
GetRelatedDocuments	$XDSAssociationStatus
	$MetadataLevel
FindDocumentsByReferenceId	$XDSDocumentEntryDocumentAvailability
	$MetadataLevel

3.18.4.1.2.3.5.1.3 Document Metadata Update Option compatibility

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

Metadata Update not implemented by Document Registry or Document Consumer

- No issues

Document Metadata Update Option implemented by both Document Registry and Document Consumer

- No Issues

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

- $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 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 the documentAvailability attribute.

1340

Document 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 rules.

1345

- 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.

1350

> *Editor: Update sub-sections under Section 3.18.4.1.2.3.7 as follows. When integrating into the Technical Framework, pay attention to notes added within tables that are highlighted in* yellow *for visibility.*

3.18.4.1.2.3.7.1 FindDocuments

…

Parameter Name	Attribute	Opt	Mult
…			
$XDSDocumentEntryStatus[7]	XDSDocumentEntry.availabilityStatus	R	M
$XDSDocumentEntryType[5]	XDSDocumentEntry.objectType	O	M
$XDSDocumentEntryDocumentAvailability	**XDSDocumentEntry.documentAvailability**	**O**	**M**
$MetadataLevel[8]	**None**	**O**	**–**

1355   …

[7]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: 4.1.5 for guidance on interpreting these results.

1360   **[8]See Section 3.18.4.1.2.3.5.1**

3.18.4.1.2.3.7.2 FindFolders

…

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

1365 …

[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: 4.1.5 for guidance on interpreting these results.

1370 **[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.**

…

3.18.4.1.2.3.7.4 GetAll

1375 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

1380 - Association objects with sourceObject or targetObject attribute matching one of the above objects **and availabilityStatus matching one of the values in the $XDSAssociationStatus parameter.**

> Note: Associations may be returned that reference objects not in the return set. For example, this could occur when:

1385 - the $XDSDocumentEntryStatus parameter is Approved and a submitted DocumentEntry has been replaced and is, therefore, Deprecated, or

- a SubmissionSet is linked to a DocumentEntry with a different Patient ID, i.e., for a mother and child.

> Document Consumers should be prepared to handle these situations.

1390

Parameter Name	Attribute	Opt	Mult
…			
$XDSDocumentEntryStatus[4]	XDSDocumentEntry.availabilityStatus	R	M
…			
$XDSFolderStatus[4]	XDSFolder.availabilityStatus	R	M
$XDSAssociationStatus	**Association.availabilityStatus**	**O**	**M**
$MetadataLevel[5]	**None**	**O**	**--**

…

4If 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/Folder object. See ITI TF-3: 4.1.5 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.

3.18.4.1.2.3.7.5 GetDocuments

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

Returns: XDSDocumentEntry objects requested

Parameter Name	Attribute	Opt	Mult
$XDSDocumentEntryEntryUUID[3]	XDSDocumentEntry.entryUUID	O[1]	M
$XDSDocumentEntryUniqueId[3,5]	XDSDocumentEntry.uniqueId	O[1]	M
$XDSDocumentEntryLogicalID[4]	**XDSDocumentEntry.logicalID**	**O[1]**	**M**
$homeCommunityId	None	O[2]	--
$MetadataLevel[6]	**None**	**O**	**--**

[1] ~~Either~~ $XDSDocumentEntryEntryUUID or $XDSDocumentEntryUniqueId **or $XDSDocumentEntryLogicalID** shall be specified. This transaction shall return an XDSStoredQueryParamNumber 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 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.

> Note: …

[3] If the Stored Query specifies a returnType of LeafClass then the Document Registry shall verify that all requested DocumentEntry objects to be returned will contain the same Patient ID. If this validation fails, an XDSResultNotSinglePatient error shall be returned and no metadata shall be returned.

4Retrieving a DocumentEntry by its logicalID attribute returns the logical DocumentEntry, which includes all versions of that DocumentEntry object. See ITI TF-3: 4.1.5 for guidance

on interpreting these results. If the Document Registry does not support the Document Metadata Update Option, this parameter shall be ignored and thus cause an
1425 **XDSStoredQueryMissingParam error because $XDSDocumentEntryEntryUUID and $XDSDocumentEntryUniqueId will be missing.**

[5]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: 4.1.5 for guidance on interpreting these
1430 **results.**

[6]See Section 3.18.4.1.2.3.5.1

3.18.4.1.2.3.7.6 GetFolders

…

1435

Parameter Name	Attribute	Opt	Mult
$XDSFolderEntryUUID[3]	XDSFolder.entryUUID	O[1]	M
$XDSFolderUniqueId[3],[5]	XDSFolder.uniqueId	O[1]	M
$XDSFolderLogicalID[4]	**XDSFolder.logicalID**	**O[1]**	**M**
$homeCommunityId	None	O[2]	--
$MetadataLevel[6]	**None**	**O**	**--**

[1] ~~Either~~ $XDSFolderEntryUUID or $XDSFolderUniqueId **or $XDSFolderLogicalID** shall be specified. This transaction shall return an XDSStoredQueryParamNumber error if ~~both~~ **more than one** of these parameters **is** ~~are~~ specified.

1440 [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 specified ~~EntryUUID or UniqueID~~ **entryUUID, uniqueId or logicalID**. See ~~ITI TF-~~
1445 ~~2a:~~**Section** 3.18.4.1.2.3.8 for more details.

[3]If the Stored Query specifies a returnType of LeafClass then the Document Registry shall verify that all requested DocumentEntry objects to be returned will contain the same Patient ID. If this validation fails an XDSResultNotSinglePatient error shall be returned and no metadata shall be returned.

1450 **[4]Retrieving a Folder by its logicalID attribute returns the logical Folder, which includes all versions of that Folder object. See ITI TF-3: 4.1.5 for guidance on interpreting these results. If the Document Registry does not support the Document Metadata Update Option this parameter shall be ignored and thus cause an XDSStoredQueryMissingParam error because $XDSFolderEntryUUID and $XDSFolderUniqueId will be missing.**

1455 **[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. See ITI TF-3: 4.1.5 for guidance on interpreting these results.**

[6]See Section 3.18.4.1.2.3.5.1

1460 ### 3.18.4.1.2.3.7.7 GetAssociations

…

Parameter Name	Attribute	Opt	Mult
$uuid	None	R	M
$homeCommunityId	None	O[1]	-
$XDSAssociationStatus	**Association.availabilityStatus**	**O**	**M**
$MetadataLevel[2]	**None**	**O**	**--**

[1]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" … >.`
1465 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 specified EntryUUID or UniqueID. See Section 3.18.4.1.2.3.8 for more details.

[2]See Section 3.18.4.1.2.3.5.1

1470 ### 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:

1475 - 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
$XDSDocumentEntryEntryUUID[3]	XDSDocumentEntry.entryUUID	O[1]	M
$XDSDocumentEntryUniqueId[3,4]	XDSDocumentEntry.uniqueId	O[1]	M
$homeCommunityId	None	O[2]	--
$XDSAssociationStatus	**Association.availabilityStatus**	**O**	**M**
$MetadataLevel[5]	**None**	**O**	**--**

1480 …

⁴Specifying 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: 4.1.5 for guidance on interpreting these results.

1485 **⁵See Section 3.18.4.1.2.3.5.1**

3.18.4.1.2.3.7.9 GetSubmissionSets

…

Parameter Name	Attribute	Opt	Mult
…			
$MetadataLevel³	**None**	**O**	**--**

1490 …

³See Section 3.18.4.1.2.3.5.1

3.18.4.1.2.3.7.10 GetSubmissionSetAndContents

…

1495

Parameter Name	Attribute	Opt	Mult
…			
$MetadataLevel⁷	**None**	**O**	**--**

…

⁷See Section 3.18.4.1.2.3.5.1

3.18.4.1.2.3.7.11 GetFolderAndContents

1500 Retrieve a Folder and its contents. The Folder object is selected either by its entryUUID or uniqueId attribute. The DocumentEntry objects returned may be constrained by their formatCode and confidentialityCode attributes. More specifically, the DocumentEntries shall be limited by the following rules:

- If the $XDSDocumentEntryConfidentialityCode parameter is present in the query, then
1505 DocumentEntries shall be returned only if they match this parameter.

- If the $XDSDocumentEntryFormatCode parameter is present in the query, then DocumentEntries shall be returned only if they match this parameter.

Returns:

- Folder identified.

1510
- DocumentEntries linked to the Folder by HasMember Associations (DocumentEntries shall pass the above rules).

- The HasMember Associations identified in the previous rule **and availabilityStatus matching one of the values in the $XDSAssociationStatus parameter.**

In the above rules, Associations are only returned if both of the objects they connect are part of
1515 the return set.

Parameter Name	Attribute	Opt	Mult
…			
$XDSFolderUniqueId[5,7]	XDSFolder.uniqueId	O[1]	--
…			
$XDSAssociationStatus	**Association.availabilityStatus**	**O**	**M**
$MetadataLevel[8]	**None**	**O**	**--**

…

[7]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
1520 **version of the metadata. By the rules of this query, the contents of each version will also be returned. See ITI TF-3: 4.1.5 for guidance on interpreting these results.**

[8]See Section 3.18.4.1.2.3.5.1

3.18.4.1.2.3.7.12 GetFoldersForDocument

1525 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 specifically, for each Association object of type HasMember that has a targetObject attribute referencing the target XDSDocumentEntry object **and availabilityStatus matching one of the**
1530 **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	**Association.availabilityStatus**	**O**	**M**
$MetadataLevel[4]	**None**	**O**	**--**

…

[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: 4.1.5 for guidance on interpreting these results.

[4]See Section 3.18.4.1.2.3.5.1

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:

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

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

  - Both sourceObject attribute and targetObject attribute reference documents AND

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

- XDSDocumentEntry objects referenced by the targetObject attribute OR the sourceObject attribute of an Association object matched above.

  Note: A side effect of the query is that the specified document is returned in the results if at least one Association is returned.

  Note: A side effect of this query is that if the document specified by the $XDSDocumentEntryUUID or $XDSDocumentEntryUniqueId parameters has no associations linking it to other documents, then no documents and no associations are returned.

See ITI TF-3: 4.1.6 Document Relationships and Associations for background.

Returns: Association objects and related XDSDocumentEntry objects

Given: An XDSDocumentEntry object and a collection of association types.

Parameter Name	Attribute	Opt	Mult
…			
$XDSDocumentEntryUniqueId[4]	XDSDocumentEntry.uniqueId	O[1]	--
…			

$XDSAssociationStatus	**Association.availabilityStatus**	**O**	**M**
$MetadataLevel[5]	**None**	**O**	**--**

…

[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. 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: 4.1.5 for guidance on interpreting these results.

[5]See Section 3.18.4.1.2.3.5.1.

3.18.4.1.2.3.7.14 FindDocumentsByReferenceId

…

Parameter Name	Attribute	Opt	Mult
…			
$XDSDocumentEntryStatus[7]	XDSDocumentEntry.availabilityStatus	R	M
$XDSDocumentEntryType[6]	XDSDocumentEntry.objectType	O	M
$XDSDocumentEntryDocumentAvailability	**XDSDocumentEntry.documentAvailability**	**O**	**M**
$MetadataLevel[8]	**None**	**O**	**--**

…

[7]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: 4.1.5 for guidance on interpreting these results.

[8]See Section 3.18.4.1.2.3.5.1

Add new sub-section under Section 3.18.4.1.2.5

3.18.4.1.2.5 Compatibility of Options

…

3.18.4.1.2.5.1 Document Metadata Update Option

When the Document Registry supports 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.**

1590

- **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 objects they reference.**

1595 **When the Document Consumer supports 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**

1600

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

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

1605

> *Update to ITI TF-2: 3.18.4.1.3 Expected Actions*
>
> *Add this text to the end of this section*

3.18.4.1.3 Expected Actions

…

1610 **The Document Consumer shall accept the metadata returned in the response and process it according to its own rules.**

If the Document Consumer supports 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)**

1615 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:

1620 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.**

1625

> ***Replace*** *the example in Section 3.18.4.1.3.3 with the following.*

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"
    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-
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"
```

1630

1635

```
                    lid="urn:uuid:08a15a6f-5b4a-42de-8f95-89474f83abdf"
        mimeType="text/xml"
1640            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
1645    Hospital^^^^^^^^^^1.2.3.4.5.6.7.8.9.1789.45</rim:Value>
                    </rim:ValueList>
                </rim:Slot>
                <rim:Slot name="creationTime">
                    <rim:ValueList>
1650                    <rim:Value>200412261119</rim:Value>
                    </rim:ValueList>
                </rim:Slot>
                <rim:Slot name="hash">
                    <rim:ValueList>
1655
        <rim:Value>4cf4f82d78b5e2aac35c31bca8cb79fe6bd6a41e</rim:Value>
                    </rim:ValueList>
                </rim:Slot>
                <rim:Slot name="languageCode">
1660                <rim:ValueList>
                        <rim:Value>en-us</rim:Value>
                    </rim:ValueList>
                </rim:Slot>
                <rim:Slot name="serviceStartTime">
1665                <rim:ValueList>
                        <rim:Value>200412230800</rim:Value>
                    </rim:ValueList>
                </rim:Slot>
                <rim:Slot name="serviceStopTime">
1670                <rim:ValueList>
                        <rim:Value>200412230801</rim:Value>
                    </rim:ValueList>
                </rim:Slot>
                <rim:Slot name="size">
1675                <rim:ValueList>
                        <rim:Value>54449</rim:Value>
                    </rim:ValueList>
                </rim:Slot>
                <rim:Slot name="sourcePatientId">
1680                <rim:ValueList>
```

```
                        <rim:Value>jd12323^^^&amp;1.2.3&amp;ISO</rim:Value>
                    </rim:ValueList>
                </rim:Slot>
                <rim:Slot name="sourcePatientInfo">
1685                    <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>
1690                    <rim:Value>PID-11|100 Main
St^^Metropolis^Il^44130^USA</rim:Value>
                    </rim:ValueList>
                </rim:Slot>
                <rim:Name>
1695                    <rim:LocalizedString charset="UTF-8" value="Sample document
1" xml:lang="en-us"/>
                </rim:Name>
                <rim:Description/>
                <VersionInfo versionName="1"/>
1700                <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"
1705                    lid="urn:uuid:ac872fc0-1c6e-439f-84d1-f76770a0ccdf"
nodeRepresentation="Education"
                    objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:Classification">
                    <rim:Slot name="codingScheme">
1710                        <rim:ValueList>
                            <rim:Value>Connect-a-thon classCodes</rim:Value>
                        </rim:ValueList>
                    </rim:Slot>
                    <rim:Name>
1715                        <rim:LocalizedString charset="UTF-8" value="Education"
xml:lang="en-us"/>
                    </rim:Name>
                    <rim:Description/>
                    <VersionInfo versionName="1"/>
1720                </rim:Classification>
                <rim:Classification classificationScheme="urn:uuid:f4f85eac-e6cb-
4883-b524-f2705394840f"
                    classifiedObject="urn:uuid:08a15a6f-5b4a-42de-8f95-
89474f83abdf"
```

```
1725                     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">
1730                         <rim:Slot name="codingScheme">
                                 <rim:ValueList>
                                     <rim:Value>Connect-a-thon
        confidentialityCodes</rim:Value>
                                 </rim:ValueList>
1735                         </rim:Slot>
                             <rim:Name>
                                 <rim:LocalizedString charset="UTF-8" value="Celebrity"
        xml:lang="en-us"/>
                             </rim:Name>
1740                         <rim:Description/>
                             <VersionInfo versionName="1"/>
                     </rim:Classification>
                     <rim:Classification classificationScheme="urn:uuid:a09d5840-386c-
        46f2-b5ad-9c3699a4309d"
1745                         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"
1750                         objectType="urn:oasis:names:tc:ebxml-
        regrep:ObjectType:RegistryObject:Classification">
                             <rim:Slot name="codingScheme">
                                 <rim:ValueList>
                                     <rim:Value>Connect-a-thon formatCodes</rim:Value>
1755                             </rim:ValueList>
                             </rim:Slot>
                             <rim:Name>
                                 <rim:LocalizedString charset="UTF-8" value="CDAR2/IHE
        1.0" xml:lang="en-us"/>
1760                         </rim:Name>
                             <rim:Description/>
                             <VersionInfo versionName="1"/>
                     </rim:Classification>
                     <rim:Classification classificationScheme="urn:uuid:f33fb8ac-18af-
1765    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"
```

```
1770                    nodeRepresentation="Emergency Department"
                        objectType="urn:oasis:names:tc:ebxml-
           regrep:ObjectType:RegistryObject:Classification">
                        <rim:Slot name="codingScheme">
                            <rim:ValueList>
1775                            <rim:Value>Connect-a-thon
           healthcareFacilityTypeCodes</rim:Value>
                            </rim:ValueList>
                        </rim:Slot>
                        <rim:Name>
1780                        <rim:LocalizedString charset="UTF-8" value="Assisted
           Living" xml:lang="en-us"/>
                        </rim:Name>
                        <rim:Description/>
                        <VersionInfo versionName="1"/>
1785            </rim:Classification>
                <rim:Classification classificationScheme="urn:uuid:cccf5598-8b07-
           4b77-a05e-ae952c785ead"
                        classifiedObject="urn:uuid:08a15a6f-5b4a-42de-8f95-
           89474f83abdf"
1790                    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">
1795                    <rim:Slot name="codingScheme">
                            <rim:ValueList>
                                <rim:Value>Connect-a-thon
           practiceSettingCodes</rim:Value>
                            </rim:ValueList>
1800                    </rim:Slot>
                        <rim:Name>
                            <rim:LocalizedString charset="UTF-8" value="Cardiology"
           xml:lang="en-us"/>
                        </rim:Name>
1805                    <rim:Description/>
                        <VersionInfo versionName="1"/>
                </rim:Classification>
                <rim:Classification classificationScheme="urn:uuid:f0306f51-975f-
           434e-a61c-c59651d33983"
1810                    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"
```

```
1815                    objectType="urn:oasis:names:tc:ebxml-
        regrep:ObjectType:RegistryObject:Classification">
                    <rim:Slot name="codingScheme">
                        <rim:ValueList>
                            <rim:Value>LOINC</rim:Value>
1820                    </rim:ValueList>
                    </rim:Slot>
                    <rim:Name>
                        <rim:LocalizedString charset="UTF-8" value="Conference
        Evaluation Note"
1825                        xml:lang="en-us"/>
                    </rim:Name>
                    <rim:Description/>
                    <VersionInfo versionName="1"/>
                </rim:Classification>
1830            <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"
1835            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>
1840                    <rim:LocalizedString charset="UTF-8"
        value="XDSDocumentEntry.patientId"
                            xml:lang="en-us"/>
                    </rim:Name>
                    <rim:Description/>
1845                <VersionInfo versionName="1"/>
                </rim:ExternalIdentifier>
                <rim:ExternalIdentifier id="urn:uuid:c3fcbf0e-9765-4f5b-abaa-
        b37ac8ff05a5"
                    lid="urn:uuid:c3fcbf0e-9765-4f5b-abaa-b37ac8ff05a5"
1850            registryObject="urn:uuid:08a15a6f-5b4a-42de-8f95-
        89474f83abdf"
                    identificationScheme="urn:uuid:2e82c1f6-a085-4c72-9da3-
        8640a32e42ab"
                    objectType="ExternalIdentifier"
1855    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"/>
```

```
1860              </rim:Name>
                  <rim:Description/>
                  <VersionInfo versionName="1"/>
              </rim:ExternalIdentifier>
          </rim:ExtrinsicObject>
1865      </rim:RegistryObjectList>
      </AdhocQueryResponse>
```

3.43 Retrieve Document Set-b [ITI-43]

…

1870

Modify ITI TF-2: 3.43.4.2.3 (Retrieve Document Set transaction)

3.43.4.2.3 Expected Actions

A Document Repository or On-Demand Document Source shall retrieve the document(s) indicated in the request.

1875 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.

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
1880 **to indicate in its error response the reason the document cannot be retrieved.**

…

3.51 Multi-Patient Stored [ITI-51]

…

Modify ITI TF-2: 3.51.4.1.2 Message Semantics as follows:

1885

3.51.4.1.2 Message Semantics

The message semantics are identical to those documented for the Registry Stored Query [ITI-18] transaction except where noted below. The following sections document the differences.

Document Source and Document Registry Actors that support the Asynchronous Web Services
1890 Exchange Option shall support Asynchronous Web Services requirements as defined in ITI TF-2: V.5.

> **Note: The $MetadataLevel parameter (see Section 3.18.4.1.2.3.5.1.1) is not included in any of the queries below because these queries enable public health use cases that only need current information and therefore do not require this parameter. In addition, adding support for this parameter would impose security risks**
1895 **and have a negative performance impact.**

Volume 3 – Cross-Transaction and Content Specifications

4.1 Abstract Metadata Model

1900 | *Update Table 4.1.3.2-1: DocumentEntry Metadata Attribute Definition to add the following rows in alphabetical order in the table*

Table 4.1.3.2-1: DocumentEntry Metadata Attribute Definition

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**

1905 | *Update Table 4.1.3.4-1: Folder Metadata Attribute Definition to add the following rows in alphabetical order in the table*

Table 4.1.3.4-1: Folder Metadata Attribute Definition

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**

1910 | *Add new Section 4.1.5*

4.1.5 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:

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

1915 **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.

A logical DocumentEntry represents a document in a repository. The logical Document Entry

1920 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).

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

1925 and version:

logicalID

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

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

1930 - 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.

- All objects with the same logicalID shall be of the same type. A logicalID shall identify a

1935 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

1940 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

1945   the first 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.

1950
- 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.

1955 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-2: 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
1960 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.

1965 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.

1970 **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)

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`
Transformation	`urn:ihe:iti:2007:AssociationType:XFRM`
Addendum	`urn:ihe:iti:2007:AssociationType:APND`
Replacement with Transformation	`urn:ihe:iti:2007:AssociationType:XFRM_RPLC`

Meaning	AssociationType
Digital Signature	`urn:ihe:iti:2007:AssociationType:signs`
Snapshot of On-Demand document Entry	`urn:ihe:iti:2010:AssociationType:IsSnapshotOf`
availabilityStatus update	**`urn:ihe:iti:2010:AssociationType:UpdateAvailabilitySta tus`**
Submit arbitrary Association	**`urn:ihe:iti:2010:AssociationType:SubmitAssociation`**

1975 ## 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

1980 **Table 4.2.3.2-1: DocumentEntry Metadata Attribute Definition (previously Table 4.1-5)**

DocumentEntry Metadata Attribute	Description	Data Type (Table 4.2.3.2.1.7-2)	Coding (Section 4.2.3.1.8)	Detail (See Section)
…				
creationTime	The time the author created the document. Shall have a single value.	DTM	ebRIM Slot	4.2.3.2.6
documentAvailabili ty	**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**

Add the following new sub-sections to Section 4.2.3.2

4.2.3.2.30 DocumentEntry.documentAvailability

Description:

1985 The status of the Document in the Document Repository. Online indicates the Document in the 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.

1990 Takes one of two values: urn:ihe:iti:2010:DocumentAvailability:Online or 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.

1995
```
<rim:Slot name="documentAvailability">
  <rim:ValueList>
    <rim:Value>urn:ihe:iti:2010:DocumentAvailability:Offline</rim:Value>
  </rim:ValueList>
2000 </rim:Slot>
```

4.2.3.2.31 DocumentEntry.logicalID

Description:

A globally unique identifier used to identify the logical entry. All versions of a DocumentEntry
2005 carry the same logicalID but different and unique entryUUIDs. If not present in a submission, 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 Section 4.1.5).

Coding:

2010 Max length is unbounded. The format of the logicalID value is UUID. LogicalID shall never be 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

2015 The initial version of a DocumentEntry could have no LogicalID present:
```
<rim:ExtrinsicObject
    id="urn:uuid:3cce0135-cedb-4a26-ba00-8698ee8dde04">
…
</rim:ExtrinsicObject>
```

2020 Or could have logicalID equal to entryUUID as shown:

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

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.

```
2030  <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

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:

Max length is unbounded.

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

```
2045  <VersionInfo versionName="2"/>
```

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

Table 4.2.3.4-1: Folder Metadata Attribute Definitions (previously Table 4.1-7)

Folder Metadata Attribute	Description	Form	Coding	Detail (See Section)
…				
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

Folder Metadata Attribute	Description	Form	Coding	Detail (See Section)
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 DocumentEntries within the Folder.	CX	ebRIM ExternalIdentifier	4.2.3.1.3
…				
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**

Add the following new sub-sections to Section 4.2.3.4

4.2.3.4.11 Folder.logicalID

Description:

2055 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 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 4.1.5).

2060 **Coding:**

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

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

2065 The initial version of a Folder could have no LogicalID present:
```
<rim:RegistryPackage
    id="urn:uuid:3cce0135-cedb-4a26-ba00-8698ee8dde04">
…
</rim:RegistryPackage>
```

2070 Or could have logicalID equal to entryUUID as shown:

```
<rim:RegistryPackage
    id="urn:uuid:3cce0135-cedb-4a26-ba00-8698ee8dde04">
    lid="urn:uuid:3cce0135-cedb-4a26-ba00-
2075 8698ee8dde04">
```

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.

2080
```
<rim:RegistryPackage
    id="urn:uuid:3cce0135-cedb-4a26-ba00-8698ee8dde04"
    lid="urn:uuid:e0985823-dc50-45a5-a6c8-a11a829893bd">
…
</rim:RegistryPackage>
```
2085

4.2.3.4.12 Folder.version

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.
2090 Subsequent versions get values of 2, 3, etc. This attribute shall be returned in query responses.

Coding:

Max length is unbounded.

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

2095
```
<VersionInfo versionName="2"/>
```

4.2.4 Success and Error Reporting

…

2100 ### 4.2.4.1 RegistryErrors Element

The following modifications are made to ITI TF-3: Table 4.2.4.1-2 Error Codes. Note the update to Note 1 below the table.

Table 4.2.4.1-2: Error Codes (previously Table 4.1-11)

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**

Error Code	Discussion	Transaction (See Note 1)
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**

2105 **Note 1:**

P = Provide and Register-b

R = Register-b

SQ = Stored Query

RS = Retrieve Document Set

2110 XGQ = Cross Gateway Query

XGR = Cross Gateway Retrieve

U = Update Document Set

4.3 Additional Document Sharing Requirements

2115 | *Update Table 4.3.1-1 and add the following note at the end of the table as shown* |
|---|

4.3.1 Submission Metadata Attribute Optionality

This section lists which metadata attributes an actor shall provide when initiating a Submission Type Transaction.

The Actor/Transaction pairs addressed by this section are as follows:

2120 **Table 4.3.1-1: Sending Actor/Transaction Pairs**

Actor	Transaction	Shortname
XDS Document Source	Provide and Register Document Set-b [ITI-41]	XDS DS
XDS Document Repository	Register Document Set-b [ITI-42]	XDS DR
XDM Portable Media Creator	Distribute Document Set on Media [ITI-32]	XDM MC
XDR Document Source	Provide and Register Document Set-b [ITI-41]	XDR DS
XDR Metadata-Limited Document Source	Provide and Register Document Set-b [ITI-41]	XDR MS
XDS On-Demand Document Source	Register On-Demand Document Entry [ITI-61]	XDS OD
XDR Document Administrator	**Update Document Set [ITI-57]**	**XDR MS**
XDS Document Administrator	**Update Document Set [ITI-57]**	**XDS DR or XDS OD (See Note 1)**

Note 1: The XDS Document Administrator shall conform to the XDS Document Repository (XDS DR) column for Stable Documents and shall conform to the XDS On-Demand Document Source (XDS OD) column for On-Demand Documents.

For each actor/transaction pair across the top of Table 4.3.1-3, and each metadata attribute row,
2125 the cell indicates the requirement for that actor when creating a submission request using the
paired transaction. The requirements are expressed through these codes:

…

*Update Table 4.3.1-3 to add the new attributes **in alphabetical order**.*

2130 **Table 4.3.1-3: Sending Actor Metadata Attribute Optionality**

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

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

Table 4.3.2.1-3: Responding Actor 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**

2135