【IHE-J コネクタソン 2012】

テストケース＋審査基準

IT 基盤分野(XDS)
技術文書

2012.07.24 Rev. 0.70

１．IHE ＸＤＳ，ＸＤＳ-Ｉテストケース概要

【対象アクタ】

Doc Source、Doc Consumer、Doc Repository、Doc Registry

【概要】

【注意事項】

オプション	テスト実施
R	テスト対象とする。
O	相互にサポートされていれば実施する。

【テストケース】

Doc Source

	DOC_SOURCE		
	Test	XDS.b	XDS-I
5	XDS.b_Doc_Source_Stores_Async	RO	
6	XDS.b_Doc_Source_Stores_Document	R	
7	XDS.b_Doc_Source_Stores_with_Proxy	R	
8	XDS.b_Error_Reporting	O	
9	XDS.b_Folder_Management	RO	
10	XDS.b_Repository_Registry_Lifecycle	RO	
11	XDSNewDocumentOffline		
12	XDS_Source_Content_Wrapped_PDF		

Doc Consumer

	DOC_CONSUMER		
	Test	XDS.b	XDS-I
1	XDS.b_Cons_homeCommunityID	R	
3	XDS.b_Consumer_QR_Async	RO	
4	XDS.b_Consumer_QR_with_Proxy	R	
5	XDS.b_Consumer_Query_Retrieve	R	
6	XDS_Embedded_Content_Wrapped_PDF		
7	XDS_Embedded_Retrieve_Document		
8	XDS_REG_REP_Retrieve_Document		
9	XDS_Repository_Retrieve_Document		
10	XDS_Source_Content_Wrapped_PDF		

Doc Repository

	DOC_REPOSITORY		
	Test	XDS.b	XDS-I
4	XDS.b_Consumer_QR_Async	RO	
5	XDS.b_Consumer_QR_with_Proxy	R	
6	XDS.b_Consumer_Query_Retrieve	R	
7	XDS.b_Doc_Source_Stores_Async	RO	
8	XDS.b_Doc_Source_Stores_Document	R	
9	XDS.b_Doc_Source_Stores_with_Proxy	R	
10	XDS.b_Error_Reporting	R	
11	XDSI_Load_DICOM_Instances		R
12	XDSI_Load_Multipart_Report		R
13	XDSI_Load_PDF_Report		R
14	XDSI_Load_Text_Report		R
15	XDSNewDocumentOffline		

16	XDS_Repository_Retrieve_Document		

Doc Registry

	DOC_REGISTRY	
	Test	XDS.b
2	XDS.b_Registry_Folder_Handling	R
8	XDS.b_Consumer_QR_Async	RO
9	XDS.b_Consumer_QR_with_Proxy	R
10	XDS.b_Consumer_Query_Retrieve	R
11	XDS.b_Doc_Source_Stores_Async	RO
12	XDS.b_Doc_Source_Stores_Document	R
13	XDS.b_Doc_Source_Stores_with_Proxy	R
14	XDS.b_Error_Reporting	R
15	XDS.b_Folder_Management	R
16	XDS.b_Repository_Registry_Lifecycle	R

【トランザクション】

1.1 Async option - Query & Retrieve a document

指示

・テストパートナーと非同期の Web サービスのテストを試みる前に、同じパートナーと(XDS.b_Consumer_Query_Retrieve)の同期の Web サービスのテストを成功させておかなければならない。
・非同期テストを始める前に、同期のテストを済ませて承認されていなければならない。このテストは、一つのインスタンスだけが要求される。
・あなたのアクタに対する非同期の Web サービスのテストに対する信用を得るため、あなたは、非同期 Web サービス上のあなたのアクタに要求されるすべてのトランザクションのテストを成功させなければならない。

説明

これは、非同期 Web サービス（ドキュメントコンシューマ→ドキュメントレジストリ、そして、ドキュメントコンシューマ→ドキュメントリポジトリ）を用いたドキュメントのクエリおよび取り出しをテストする。
非同期 Web サービスのための SOAP ヘッダ属性の評価を可能にするため、NIST プロキシ—を通じたテストを実施する。. NIST プロキシ—（技術的には、リバースプロキシ—）は、非同期の Web サービストランザクションを受け付けることができる各ベンダアクタのフロントで実行されている。構成情報は、Bill Majurski.から入手可能である。

・プロキシ—を利用するために、送り出すシステム（コンシューマ）を構成する。これは、メッセージが、nist1:xxxx に送られることを意味している（ここで、xxxx は、NIST によって割り当てられたポートである）。しかし、HTTP ヘッダおよび WS:To ヘッダは、全部の目標となるベンダアクタのエンドポイントを指す。
・テストを始める前に、あなたの構成および NIST プロキシ—の準備状態を NIST の Bill Majurski に確認すること。あなたの WS: Headers の評価は、プロキシ—コンソールから行われる。

データの正しい交換を示すためにパートナーと直接テストを行う。

評価

- 交換の非同期の状況を検証するためにシステムのログを見せる用意をする。
- 正しいアクタの操作を示すために準備をする。

このテストに関連する統合プロファイル／アクタ:

Integration Profile	Actor	Profile Option	Option	Meta Test
XDS.b	DOC_CONSUMER	ASYNCH_WEB_SVCS_EXCHANGE	RO	
XDS.b	DOC_REGISTRY	ASYNCH_WEB_SVCS_EXCHANGE	RO	
XDS.b	DOC_REPOSITORY	ASYNCH_WEB_SVCS_EXCHANGE	RO	

テストシナリオ:

Step	Trans.	Msg Type	Return	From Actor	To Actor	Step Description	Opt.
5	NULL			--	--	テスト説明での指示に従って準備をする。	R
10	ITI-18			DOC_CONSUMER	DOC_REGISTRY	ドキュメントコンシューマは、一人の患者に対するドキュメントリストを得るために、ドキュメントレジストリに対してストアードクエリを送信する。あなたは、レジストリ／リポジトリペア内に存在するどの患者でも使うことができる。	R
20	ITI-43			DOC_CONSUMER	DOC_REPOSITORY	ドキュメントコンシューマは、ドキュメントリポジトリからのドキュメントを取り出す。	R

1.2 Query and Retrieve document with NIST Proxy

指示
各 XDS.b ドキュメントコンシューマ、レジストリ、およびリポジトリは、このテストの一つのインスタンスを実行することが要求される。
レジストリ及びリポジトリは、複数のドキュメントコンシューマを支援するために一回以上、実行しなければならない。
TLS は、このテストでは使用されない。

説明、評価
NIST ツールは、あなたなのトランザクションの文脈で、評価を実施することになる。

このテストに関連する統合プロファイル／アクタ:

Integration Profile	Actor	Profile Option	Option	Meta Test
XDS.b	DOC_CONSUMER	NONE	R	
XDS.b	DOC_REGISTRY	NONE	R	
XDS.b	DOC_REPOSITORY	NONE	R	

テストシナリオ:

Step	Trans.	Msg Type	Return	From Actor	To Actor	Step Description	Opt.

1.3 XDS.b Consumer Query Retrieve

指示
このテストを行う前に、Doc Registry および Repository は、テスト **XDS.b_*_Do_This_First** をパスしなければならない。
TLS 通信を利用してこのテストを実施すること。
Doc Sources によって登録されたドキュメントを見つけるために、ドキュメントコンシューマ は、Connectathon Images/CDs/Objects page を使うべきである。
Objects Information.上の Objects to Render'タブを参照。
このテストの ATNA 監査ログを受け取る監査レコードリポジトリ（ARR）の名前を記録すること。

説明
XDS_Consumer_Query_Retrieve テストは、XDS ドキュメントコンシューマの観点から、XDS プロファイルを試行する。
ドキュメントリストを得るため、XDS レジストリに患者 ID で特定のクエリを出す。
XDS コンシューマは、ドキュメントを取り出し、コネクタソン審査員にそれらを表示する。
テストされるドキュメントコンシューマは、ドキュメントレジストリの少なくとも1つのクエリを発行することを要求する。そのクエリの形式は、規定されていない。テクニカルフレームワークは、多くのオプションを定義している。
評価のためには、コネクタソンモニターは、クエリ、およびドキュメントコンシューマアクタでの取り出しのプロセスを観察すべきである。

評価
ドキュメントコンシューマ上で、コネクタソンの審査員は、リアルタイムでクエリおよび取り出し、および。両方のトランザクションが TLS 通信を用いて実施されたことのログやまたは他のエビデンスを観察することになる。

このテストに関連する統合プロファイル／アクタ:

Integration Profile	Actor	Profile Option	Option	Meta Test
XDS.b	DOC_CONSUMER	NONE	R	
XDS.b	DOC_REGISTRY	NONE	R	
XDS.b	DOC_REPOSITORY	NONE	R	

テストシナリオ:

Step	Trans.	Msg Type	Return	From Actor	To Actor	Step Description	Opt.
10	NULL			--	--	1 つ以上の患者を見つけるため、選択されたレジストリ及びリポジトリに登録されたドキュメントを持 Connectathon -> Connectathon Objects のもとで Kudu を調べる。	R
20	ITI-19			--	--	認証ノード: TLS 通信の開始	R
30	ITI-18			DOC_CONSUMER	DOC_REGISTRY	ドキュメントコンシューマは、一人の患者に対して、ドキュメントリストを得るために、ドキュメントレジストリに対してストアードクエリを送信する。	R
40	ITI-43			DOC_CONSUMER	DOC_REPOSITORY	ドキュメントコンシューマは、ドキュメントリポジトリからドキュメントを取り出す。	R
80	NULL			--	--	自分のログファイルを開き、トランザクションが TLS 通信を使って実施されたことのエビデンスを示すログ情報を検索する。XDS_Consumer_Query_Retrieve_NNN と呼ぶテキストファイル（又は他のドキュメント）を生成する、ここで、NNN は、Kudu テスト番号である。このファイルをコネクタソン審査員に渡すまで保持する。	R

1.4 XDS.b Document Source Stores Document

指示
Doc Source, Registry, および Repository は、このテストを行う前に、**XDS.b_*_Do_This_First** のテストをパスしていなければならい。

注1:独自のドキュメントを生成しないドキュメントソースシステム(ミドルウェアとして動作する)は、すべての XDS テストで定義された特定の患者に対するドキュメントを生成するように準備してコネクタソンに望まなければならない。
ドキュメントソースシステムは、テスト指示によって定義された患者を利用して、オンデマンドで、ドキュメントを生成することが要求される。

このテストの ATNA 監査ログを受け取る監査レコードリポジトリ(ARR)の名前を記録すること。

これは、つぎの3つのシステムの手当てが必要であり、難しいテストである:
Document Source
Document Repository
Document Registry
Document Source から始めたリンク又は Document Registry で終わるリンクをはじめに試すかの指針は与えない。結局は、エンドからエンドの操作を試さなければならない。
このテストの終了時には、格納されたドキュメントについての情報を格納するため Kudu の共通の場所を更新してほしい。
このテストは、Document Source が、審査員がレジストリおよびリポジトリの中のソースドキュメントをより容易に発見できるような患者名を使用することを要求する。

説明
XDS_Doc_Source_Stores_Document のテストは、ITIトランザクション41及び42のテストである。Document Source は、少なくとも、1つのドキュメントを供給し、Document Repository へそのドキュメントを提出する。このテストは、1 つのサブミッションセットおよび1つのドキュメントをカバーする。Document Source へのオプションは、異なるテストでカバーされる。ITI-41 は、ITI-42: Register Document Set に強く結びついている。Document Source が、オリジナルのドキュメントを提出したとき、Document Repository は、次に、Document Registry にそのドキュメントを登録する。
このテストは、TLS 通信を用いてテストを実施すべきである。このことは、コネクタソンの開始時に、TLS の作業を行っておく必要があることを意味する。
このテストに対しては、ATNA のログをとることは要求されないが、TLS 通信を使う必要がある。

このテストを完了したとき、Doc Consumers が、Connectathon Images/CDs/Objects の下のドキュメントを見つけられるように Kudu ツールの参照情報を記録しなければならない。このページへのリンクは、*Objects Information. である。*

提出には、いくつかの値の入力を促される。コネクタソンの審査員は、Submission Set uniqueID で参照されるドキュメントを見つけることができない場合には、テストは失敗とみなさる。コネクタソンの審査員は、このレジストリまたは他のレジストリの中に Submission Set uniqueID の複数インスタンスを発見した場合には、そのテストは、失敗とみなさる。このコネクタソンの間、すべてのサブミッションに対するユニークな Submission Set uniqueID を生成する必要がある。審査員は、このテストを走らせたエビデンス/ログを求めたり。または、立会いのもと再度テストを実行すること求めることができる。
このテストを終了したとき、このテストのこのインスタンスの Kudo のチャットウインドウの中にログ情報/エビデンスを記録すべきである。これにより、審査員にあなたのログデータを素早く見せることができるようになる。

評価
評価は、2 つのフェーズで行われる。最初のフェーズは、一人のコネクタソン審査員が、システムに訪れ、ログファイルを調べてトランザクションが TLS を用いて実行されていることのエビデンスをチエックする。コネクタソン審査員は、テストを再実行することを求めることができる。ログの要求内容は、 以下のとおり(ITI TF-2:3.41.7 & 3.42.7)。

ITI-42 (Provide & Register)に対しては、(他のフィールドの)ドキュメントメタデータから患者 ID および submissionSet uniqueID を含む"Export" event のログを取る。
ITI-42 (Provide & Register)に対しては、(他のフィールドの)ドキュメントメタデータから患者 ID および submissionSet uniqueID を含む"Import" event のログを取る。
ITI-41 (Register Doc)に対しては、(他のフィールドの)ドキュメントメタデータから患者 ID および submissionSet uniqueID を含む"Export" event のログを取る。
ITI-41 (Register Doc)に対しては、(他のフィールドの)ドキュメントメタデータから患者 ID を含む"Import" event のログを取る。

セキュリティ要件が確認された後は、審査員は、状態を部分的に確認済みに設定する。
評価の 2 つ目のパートは、Submission Set ID が見つかるかどうか、その値が複数の値であるかどうかを決定するために Registry に対して再びクエリ実行することである。
これは、登録されたドキュメントの検索によって確かめられる。

URI は、TLS (https://)を示していなければならない。

最後に、Document Source システムで生成されたドキュメントは、が IHE コンテンツプロファイルの一つに準拠していないならば、コネクタソン審査員は、このシステムで生成されたドキュメントは、そのような一般的な電子フォーマットのラッピングサポートを含む公開されたヘルスケア

標準に準拠することを確認する。非構造データ、テキストファイル、固有のドキュメントフォーマット（例えば、RTF）、または、イメージ、は、標準的なドキュメント形式内にカプセル化されるべきである(Ref ITI TF-1: 10.4.2)。

このテストに関連する統合プロファイル／アクタ:

Integration Profile	Actor	Profile Option	Option	Meta Test
XDS.b	DOC_REGISTRY	NONE	R	
XDS.b	DOC_REPOSITORY	NONE	R	
XDS.b	DOC_SOURCE	NONE	R	

テストシナリオ:

Step	Trans.	Msg Type	Return	From Actor	To Actor	Step Description	Opt.
10	NULL			--	--	ドキュメントレジストリ、PIX マネジャおよび PDQ サプライヤは、コネクタソンの基本情報スプレードシートから患者 ID で事前登録される。それらの基本情報は、姓として XDS の Doc Source のシステム名を含む。例えば、Ehrabc^Mary。	R
20	NULL			DOC_SOURCE	--	Document Source は、コネクタソンでの基本情報および作られた患者名から、患者 ID を利用する新しいドキュメントを生成する。ドキュメントのそれらの値を使用して、最低限、患者名を提出する。1 つのドキュメントを生成し、異なるリポジトリにまき散らしてはならない。	R
30	ITI-19			DOC_SOURCE	DOC_REPOSITORY	認証ノード: TLS 通信を開始。	R
40	ITI-41			DOC_SOURCE	DOC_REPOSITORY	Document Source は、あなたの Document Source's の後ろにベンダ名が付けられた患者に対する Document Repository に一つのドキュメントを送信する。	R
42	ITI-20			--	--	Doc Source は、"Export" event、 Doc Repository は、"Import" event のログを取る。	R
45	ITI-19			DOC_REGISTRY	DOC_REPOSITORY	認証ノード: TLS 通信を開始。	R
50	ITI-42			DOC_REPOSITORY	DOC_REGISTRY	Document Repository は、Document Registry にドキュメントを登録する。	R
52	ITI-20			--	--	Doc Repository は、"Export" event、 Doc Registry は、"Import" event.のログを取る。	R
55	NULL			--	--	チャットウインドウに SubmissionSet.unique ID の値を貼りつける。それをラベル付けする。例えば、SSID: 1.2.3.4.5. 審査員は、このテストを検証するためにその値が必要である。	R
60	ITI-41			DOC_SOURCE	DOC_REPOSITORY	（オプション）Document Source は、1つのサブミッションで複数のドキュメントを送付することが可能である。 ・今、それを実施する。（Document Sources は、この機能をもつことを要求はされない。しかし、Document Registries および Repositories は、これをサポートしなければならない）。	O
70	ITI-42			DOC_REPOSITORY	DOC_REGISTRY	（オプション）Document Repository は、Document Registry にドキュメントを登録する。	O
80	NULL			--	--	あなたのログファイルを開いて、TLS 通信を用いてトランザクションが実施されたことのエビデンスを示すログ情報を取り出しなさい。	R
90	NULL			--	--	このステップは、Document Consumers にあなたのドキュメントのテストの支援を得る。Web ブラウザを用いて Kudu にアクセスする。Connectathon -> Connectathon. Objects。あなたが格納したドキュメント上の情報を追加する。それには、Patient Name, Patient ID, Document Source, Document Repository, Document Registry, Document Type & Submission Set ID を含むことになる。	R
100	NULL			--	--	NIST ツールキットを利用して、あなた自身のテストを検証しなさい。ツール上では、Connectathon Tools.の下で、このテストへのリンクを見つけなさい。	R

						このテストのチャットウインドウで、ログの印"Pass"をつけて、コピー／ペーストしなさい。 このテストを、あなたが成功を得るまで"ready to be verified"にしないこと。	
130	NULL			--	--	XDS_Error_Reporting.を終了するために、直ちに、実行することが許される。	O

1.5 Async option - Provide & Register a document

指示

・テストパートナーと非同期の Web サービスのテストを試みる前に、同じパートナーと(XDS.b_Doc_Source_Stores_Document)の同期の Web サービスのテストを成功させておかなければならない。
・非同期テストを始める前に、同期のテストを済ませて承認されていなければならない。このテストは、一つのインスタンスだけが要求される。
・あなたのアクタに対する非同期の Web サービスのテストに対する信用を得るため、あなたは、非同期 Web サービス上のあなたのアクタに要求されるすべてのトランザクションのテストを成功させなければならない。

説明

これは、非同期 Web サービス Doc Source-->Doc Repository-->Doc Registry を用いたドキュメントのクエリおよび取り出しをテストする。
非同期 Web サービスのための SOAP ヘッダ属性の評価を可能にするため、NIST プロキシ—を通じたテストを実施する。. NIST プロキシ—(技術的には、リバースプロキシ—)は、非同期の Web サービストランザクションを受け付けることができる各ベンダアクタのフロントで実行されている。構成情報は、 Bill Majurski.から入手可能である。

・プロキシ—を利用するために、送り出すシステム(Source, Repository)を構成する。これは、メッセージが、nist1:xxxx に送られることを意味している(ここで、xxxx は、NIST によって割り当てられたポートである)。しかし、HTTP ヘッダおよび WS:To ヘッダは、全部の目標となるベンダアクタのエンドポイントを指す。
・テストを始める前に、あなたの構成および NIST プロキシ—の準備状態を NIST の Bill Majurski に確認すること。あなたの WS: Headers の評価は、プロキシ—コンソールから行われる。

データの正しい交換を示すためにパートナーと直接テストを行う。

評価

- 交換の非同期の状況を検証するためにシステムのログを見せる用意をする。
- 正しいアクタの操作を示すために準備をする。

このテストに関連する統合プロファイル／アクタ:

Integration Profile	Actor	Profile Option	Option	Meta Test
XDS.b	DOC_REGISTRY	ASYNCH_WEB_SVCS_EXCHANGE	RO	
XDS.b	DOC_REPOSITORY	ASYNCH_WEB_SVCS_EXCHANGE	RO	
XDS.b	DOC_SOURCE	ASYNCH_WEB_SVCS_EXCHANGE	RO	

テストシナリオ:

Step	Trans.	Msg Type	Return	From Actor	To Actor	Step Description	Opt.
5	NULL			--	--	テスト説明での指示に従って準備をする。	R
10	ITI-41			DOC_SOURCE	DOC_REPOSITORY	Document Source は、Document Repository に単独のドキュメントを送信する。Registry.で知られている任意の患者 ID を使用する。	R
20	ITI-42			DOC_REPOSITORY	DOC_REGISTRY	Document Repository は、Document Registry にドキュメントを登録する。	R
30	NULL			--	--	このテストのために Kudu のチャットウインドウに Submission Set uniqueID を記録する。	R

1.6 Store a document through the NIST Proxy

指示
各 XDS.b の Document Source, Registry and Repository は、このテストの1つのインスタンスを実行することが要求される。
Registries および Repositories は、複数の Doc Sources.を支援するために一度以上、実行しなければいけないかもしれない。
TLS は、このテストでは、使用しない。

説明、評価
NIST ツールは、あなたのトランザクションのコンテンツに関しての評価を実行する。

このテストに関連する統合プロファイル／アクタ:

Integration Profile	Actor	Profile Option	Option	Meta Test
XDS.b	DOC_REGISTRY	NONE	R	
XDS.b	DOC_REPOSITORY	NONE	R	
XDS.b	DOC_SOURCE	NONE	R	

テストシナリオ：

Step	Trans.	Msg Type	Return	From Actor	To Actor	Step Description	Opt.

1.7 XDS.b Error Reporting

指示
Document Registry および Document Repository システムは、このテストを行う前に、**XDS_Doc_Source_Stores_Document** のテストをかんりょうしておくべきである。
Document Registry および Document Repository システムは、このテストの1つのインスタンスを実行すべきである。
Document Source は、このテストで補助の役割を果たすので、オプショナルなアクタとして認識されている。

記述
これは、ITI-42 Register Document Set-b に対して報告されたエラーをテストする。Registry actor および Repository アクタは、ITI TF-2: 4.1.13 の中で定義されたコードセットを使用してエラーを報告する必要がある。これは、エラー条件の小さなサブセットをテストする、しかし、コネクタソン審査員は、コネクタソンのテスト期間中に発生した他のエラー条件も監視し、それらの場合も同様にエラーコードをチックすることがある。それらの条件で、適切にエラーを報告することが成功したか、失敗したかが、このテストの一部である。審査員は、それらのテストを実行した際のエビデンス／ログを要求するか、立会いの下、再実行を求める。テストを終了したとき、このテストのこのインスタンスに対する Kudu チャットウインドウにログ情報／エビデンスを記録すべきである。これで、審査員にあなたのログデータを素早く見せることができる。

評価
コネクタソン審査員は、2つのエラー条件を評価する。

1. ドキュメントソースが、Registry.で知られていない患者に対するドキュメントを提出した。
Registry により報告されたエラーは、このようなものであるべきである（*errorCode の値は、マッチしなければならない、しかし、codeContext のコンテンツは、テクニカルフレームワークでは必須ではない、そして、実装依存であってもよい;これは、可能な値のサンプルを含む*）
errorCode="XdsUnknownPatientId"
codeContext="Patient ID is not known to the registry"
location=""
severity="Error"/>

2.（オプショナルなステップ）Document Repository は、Document Registry に接続するのに失敗した。
Repository により報告されたエラーは、このようなものであるべきである（*errorCode の値は、マッチしなければならない、しかし、codeContext のコンテンツは、テクニカルフレームワークでは必須ではない、そして、実装依存であってもよい;これは、可能な値のサンプルを含む*）
<>
xmlns="urn:oasis:names:tc:ebxml-regrep:registry:xsd:2.1" status="Failure">
<>
errorCode="XdsRegistryNotAvailable"
codeContext="Unable to connect to the Registry to register document"
location=""
severity="Error"/>

このテストに関連する統合プロファイル／アクタ:

テストシナリオ：

Step	Trans.	Msg Type	Return	From Actor	To Actor	Step Description	Opt.
10	NULL			--	--	Document Registry は、次のような患者に対する Patient Identity Feed を受け付けないことに注意。これは、ここでテストするエラー条件である。	R
20	NULL			--	--	Document Source は、新しい Patient ID および名前として Error^Reporting をもつ新しいドキュメントを生成する。ドキュメント中のそれらの値を使用する（最低、患者名および ID）。	R
30	ITI-19			--	--	Authenticate Node: Begin TLS communication. 認証ノード：TLS 通信を開始。	R
40	ITI-41			DOC_SOURCE	DOC_REPOSITORY	Document Source は、患者 Error^Reporting に対して、Document Repository に1つの単体のドキュメントを送信する。	R
50	ITI-42			DOC_REPOSITORY	DOC_REGISTRY	Document Repository は、Document Registry にドキュメントを登録する。この Registry は、これは、知らない患者であるので、エラーを報告すべきである。エラー返答のコピーを後の評価のため取っておくこと。	R

55	NULL			--	--	Document Source は、Repository から返されたエラーの受け取りを適切に扱うべきである。このステップは、評価の間に行なわれた判断を適切に導く認識として“オプショナル”と明記されている。	O
60	NULL			--	--	Document Registry は、このテストで終了し、他のテストに移ることができる。	R
70	NULL			--	--	つぎのステップは、Registry に接続できない Repository のエラー条件をテストする。	O
80	NULL			--	--	Document Repository は、このエラー条件を引き起こす存在しない Registry を設けるべきである。	O
90	ITI-41			DOC_SOURCE	DOC_REPOSITORY	Document Source は、患者 Error^Reporting として Document Repository に他の単体のドキュメントを送る。	O
100	ITI-42			--	--	Document Repository は、Document Registry.にドキュメントを登録することを試みる。Repository は、Registry に接続できなかった場合は、エラーを報告すべきである。後の評価のため、チャットウインドウでエラー応答のコピーを保存しておくこと。	O

1.8 Test folders with NIST toolkit

Special Instructions
When you are ready to exercise your Registry's folder-handling functionality, mark this test 'Ready to be Verified'. This will trigger the XDS monitors to run a set of tests from the NIST tool set against your Registry.
To run the XDS.b_Registry_Folder_Handling test, you need to inform us of a Patient ID that your Registry will accept. Paste the patient ID value into the chat window for this test.. Then mark the test as 'Completed'. As part of the validation, Bill will run the folder tests from his toolkit. If successful the monitor will mark the test 'Verified'.
After you run this test, Doc Registries should find a Document Source that support the Folder Management option and move on to test XDS.b_Folder_Management.

Description
You must pass the following tests from the NIST XDS toolkit:
11999 Accept Create Folder
12000 Accept Create Folder with Initial Document
12001 Add new document to existing folder
12326 Add Existing document to existing folder
12002 Reject Add Document to Folder - Patient ID does not match
12327 Accept Document Replace, Document in Folder
12323 Folder lastUpdateTime

Evaluation
An XDS Monitor will evaluate this test.
If your NIST tests pass, you will pass this connectathon test.

Integration Profiles / Actors Concerned by the test :

Integration Profile	Actor	Profile Option	Option	Meta Test
XDS.b	DOC_REGISTRY	NONE	R	

Test script

Step	Trans.	Msg Type	Return	From Actor	To Actor	Step Description	Opt.
10	NULL			DOC_REGISTRY	--	See test description above to identify a Patient ID which your Registry will accept. Paste that into the chat window. When you are ready to exercise your Registry's folder-handling functionality, mark this test 'Ready to be Verified'. This will trigger a set of tests from the NIST tool set to be run against your Registry. There are no steps you must execute yourself.	R

1.9 Your first test: Receive a document submission from the NIST toolkit

指定

Document Repository として、他の XDS.b テストを始める前にこのテストをパスすべきである。
このテストに対して、よく知っている患者を使用する：**FARNSWORTH^STEVE patient id 101^&1.3.6.1.4.1.21367.2010.1.2.300&ISO** すべての Doc Registries の上に、コネクタソン患者基本情報の一部としてロードされている。
XDS ツールのローカルな **nist1** インスタンスを使用する。

記述

このテストでは、NIST ツールキットが、あなたの Repository にドキュメントを提出する。NIST ツールキットからのサブミッションを受けられるように、あなたの Repository を準備したのち、このテストを、"Ready to be verified"としてマークを付ける。これは、あなたの Repository.に対するツールキットからのテストを起動するきっかけとなる。

評価

コネクタソン審査員は、あなたのテストが TLS を使って実行されたこと、あなたのサブミッション結果がパスしたことを指名す NIST ツールキットのアプトプットに対するチャットウインドウを点検する。

このテストに関連する統合プロファイル／アクタ:

Integration Profile	Actor	Profile Option	Option	Meta Test
XDS.b	DOC_REPOSITORY	NONE	R	

Test script

Step	Trans.	Msg Type	Return	From Actor	To Actor	Step Description	Opt.
10	ITI-19			--	--	Access the NIST toolkit and find this test in the Connectathon Tools menu. Select TLS.	R
20	NULL			--	--	The NIST toolkit will provide-and-register a document to your Repository. Paste the output of the tool indicating "Pass" into the chat window for this test, then mark the test "Ready to be verified".	R

1.10 Doc Replacement, Addendum & Transformation

Special Instructions
What was previously known as the 'Document Lifecycle option' for XDS Document Source actors and Integrated Document Source/Repository actors is now split into separate options: 'Document Replace', 'Document Addendum' and 'Document Transformation'.
In this test, we use a Document Source that supports one or more these options. The Repository is an optional actor in this test because it is needed to help with the test, but no Repository function is evaluated.

Description
In the test steps below, we will use one patient created in the Geneva Tool or by a Patient Identity Source. The patient name is a composite of the Document Source and Document Repository actors using the following convention:

- SOURCE^REPOSITORY^L: L stands for Document Life Cycle Management

Evaluation
It is likely that a Connectathon monitor will be assigned to find participants who support these options, visit these system to run the tests, and record pass/fail on a spreadsheet.
Results will depend on which options the Doc Source supports.
Connectathon monitor will examine log files and database entries of the Document Repository and Document Registry to look for evidence of documents. Alternatively, use a Document Consumer to retrieve the documents. Monitor may ask the Doc Source to exercise the options again, real-time.

- For the first document, in the append and transform case, the status of the original document remains unchanged (approved).
- For the second document, in the replacement case, the status of original document, and the addendum and replacement, should be deprecated.' The replacement version of the document is approved, and is returned in response to a retrieve.

Integration Profiles / Actors Concerned by the test :

Integration Profile	Actor	Profile Option	Option	Meta Test

Test script

Step	Trans.	Msg Type	Return	From Actor	To Actor	Step Description	Opt.

２．メタデータ定義仕様

１　概要

XDS メタデータは、ドキュメントエントリ、サブミッションセット、フォルダから構成されている。図1は、XDS メタデータの構成を示す。ここでは、メタデータの詳細な形式を定義する。

・同じフォルダに登録する複数のドキュメントエントリをまとめて、サブミッションセットを構成し登録の単位とする。

・ドキュメントエントリごとに、文書ID，地域患者IDなどのドキュメントの内容を反映したメタデータをつける。

・登録ドキュメントの種別は、ドキュメントエントリのメタデータ classCode、typeCode、MIME タイプ及び、フォーマットコードで識別する。

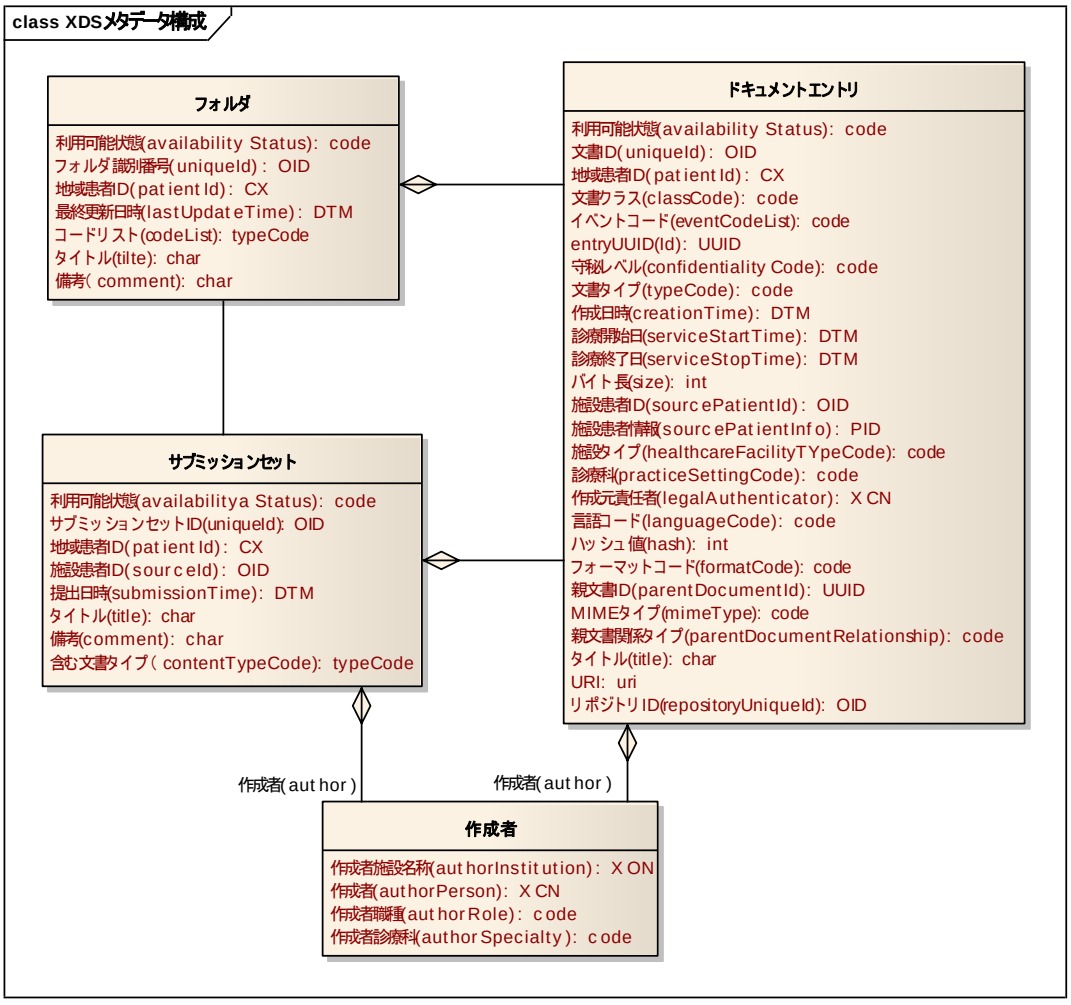

図１　XDS のメタデータ構成図

XDS メタデータの定義、XML 形式の規定および XML インスタンスの例を挙げる。詳細な形式は、表 1 の標準を参照。

表１　メタデータ定義で参照する標準の一覧

ebRIM	OASIS/ebXML Registry Information Model v3.0
ebRS	OASIS/ebXML Registry Services Specifications v3.0
IHE ITI	Registry Stored Query Transaction for XDS Profile [ITI-18], Trial Implementation Version

IHE ITI	vol. 2 (ITI TF-2): Transactions, Revision 4.0
IHE ITI	Cross-Enterprise Document Sharing-b (XDS.b) Draft for Trial Implementation

適合性の区分を、表2に示す。

表２　メタデータ定義における適合性の区分　（参考）

項目	適合性基準	備考
ebRIM および ebRS（スキーマ）	（基準 A）	
固定値	（基準 A）	
指定された語彙が基準 A の部分	（基準 A）	
指定された語彙が基準 B の部分	（基準 B）	該当箇所のボキャブラリの拡張が可能
指定された語彙が基準 C の部分	（基準 C）	該当なし

表３　メタデータ定義表の項目

項目	内容
XPath	XML スキーマ上の位置を示す。
要素名/属性名	XML の要素名または属性名を示す（属性名は、@ではじまる文字列）。
内容	要素名/属性名の説明
型/語彙	要素または属性の値のデータ型または語彙を示す。データ型は、以下のとおり。 ①一般：char（文字列）, int（数値）, code（コード値） ②ID 関係：UUID, OID, URI ③HL7 関係：DTN（日付）,CX（患者 ID）,XCN（名前）,XON（組織名）,PID（患者基本）
設定	値の設定に関する制約を示す。以下の意味をもつ。 ・生成：処理系が自動で値を設定する。 ・固定：固定の UUID、コード値、文字列を指定する。 ・R：値が、必須である。 ・O：値は、オプションである。
多重	XML スキーマとしての取り得る値の多重度を示す。

注）以下の XML インスタンスの例示では、固定値は下線で表示する。また、枠付きの値は、適切な値がセットされることを示す。

項目	必要性	項目	必要性
author	R2	entryUUID	Cg
authorInstitution	R2	homeCommunityId	Cx
authorPerson	O	intendedRecipient	O
authorRole	R2	patientId	R
authorSpecialty	R2	sourceId	R
availabilityStatus	Cg	submissionTime	R
comments	O	title	O
contentTypeCode	R	uniqueId	R
contentTypeCodeDisplayName	R		

凡例　R:必須　R2:明らかであればできるだけ記入する　O:任意　Cg:レジストリにて設定（必須）
Cx:レジストリにて設定（任意）

２　ドキュメントエントリ（DocumentEntry）

２．１　repositoryUniqueID, entryUUID, availabilityStatus, mimeType 及び title

ドキュメントエントリを、ebRIM3.0 の外部オブジェクト（ExtrinsicObject）として登録する。外部オブジェクトの状態（ライフサイクル）が管理される。

表３　DocumentEntry – repositoryUniqueID, entryUUID, availabilityStatus, mimeType 及び title のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/			
ExtrinsicObject	外部オブジェクト			
@id	外部オブジェクトの識別子（**entryUUID**）を指定。 （内部で、UUID に変換される）	char (UUID)	R	1..1
@objectType	ebXML のオブジェクトタイプを UUID で指定。 "**urn:uuid:7edca82f-054d-47f2-a032-9b2a5b5186c1**" に固定。	UUID	固定	1..1
@status	・**availabilityStatus** を指定。 ・サブミッション処理後に、承認（Approved）状態にセットされる。 （ドキュメントソースからの提出・登録時には、指定不要）。 ・ドキュメントのオーナは、廃止（Deprecated）にできる。	UUID	生成	1..1
@mimeType	外部オブジェクトの **A-mimeType**(7.2.10)を指定。	code (基準 A)	R	1..1
/ExtrinsicObject/Name				0..1
LocalizedString	日本語での表記。			1..1
@value	ドキュメントのタイトル（**title**）を記載。	char	O	1..1
/ExtrinsicObject/Slot				1..1
@name	スロットの名称。"**repositoryUniqueId**"に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	ドキュメントリポジトリを一意に識別する ID **(repositoryUniqueId)**を指定。	OID	R	1..1

```
< RegistryObjectList>
  < rim:ExtrinsicObject
    id="theDocument"
    objectType= "urn:uuid:7edca82f-054d-47f2-a032-9b2a5b5186c1"
    status="Approved の UUID"
    mimeType="text/xml">
    <rim:Name>
      <rim:LocalizedString value="title"/>
    </rim:Name>
    <rim:Slot name ="repositoryUniqueId">
      <rim:Valuelist>
        <rim:Value>repository の OID</rim:Value>
      </rim:ValueList>
    </rim:Slot>
      …

  </rim:ExtrinsicObject>
</RegistryObjectList>
```

図２　DocumentEntry – repositoryUniqueId, entryUUID, availabilityStatus, mimeType 及び title のメタデータ例

２．２　uniqueId（文書 ID）

ドキュメントエントリをレジストリ内で一意に識別する ID を(ドキュメントソース側で)指定する。

表４　DocumentEntry - uniqueId（文書 ID）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ExtrinsicObject/			
ExternalIdentifier	外部識別子として指定。			1..1
@id	自身のオブジェクトを指す UUID を指定。	UUID	R	1..1
@identificationScheme	外部識別子 uniqueId を指す UUID を指定。 "**urn:uuid:2e82c1f6-a085-4c72-9da3-8640a32e42ab**"に固定。	UUID	固定	1..1
@value	ドキュメントエントリを識別する OID。	OID	R	1..1
@registryObject	DocumentEntry(ExtrinsicObject)に割り当てられた UUID を指定。	UUID	R	1..1
/ExternalIdentifier /Name				0..1
LocalizedString	日本語での表記。			1..1
@value	"**文書 ID (XDSDocumentEntry.uniqueId)**"に固定	char	固定	1..1

```
<rim:ExternalIdentifier
   id="e1"
   identificationScheme= "urn:uuid:2e82c1f6-a085-4c72-9da3-8640a32e42ab"
   value="1.3.6.1.4.1.21367.2005.3.7^11379"
   registryObject="theDocument">
   <rim:Name>
      <rim:LocalizedString value="文書 ID(XDSDocumentEntry.uniqueId)"/>
   </rim:Name>
</rim:ExternalIdentifier>
```

図３　DocumentEntry - uniqueId（文書 ID）のメタデータ例

２．３　patientId（地域患者 ID）

ドキュメントエントリに関する地域患者IDを指定する。

表５　DocumentEntry - patientId（地域患者 ID）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ExtrinsicObject/			
ExternalIdentifier	外部識別子として指定。			1..1
@id	自身のオブジェクトを指す UUID を指定。	UUID	R	1..1
@identificationScheme	外部識別子 uniqueId を指す UUID。 "**urn:uuid:58a6f841-87b3-4a3e-92fd-a8ffeff98427**"に固定。	UUID	固定	1..1
@value	PIX で発行された地域患者 ID を指定。 ・識別子発行機関 ID（PIX センタを指す OID） ・地域患者 ID（PIX で管理発行）	CX	R	1..1
@registryObject	DocumentEntry(ExtrinsicObject)に割り当てられた UUID を指定。	UUID	R	1..1
/ExternalIdentifier /Name				0..1
LocalizedString	日本語での表記。			1..1
@value	"**地域患者 ID (XDSDocumentEntry.patientID)**"に固定	char	固定	1..1

```
<rim:ExternalIdentifier
   id="e2 "
   identificationScheme= "urn:uuid:58a6f841-87b3-4a3e-92fd-a8ffeff98427"
   value="6578946^^^&amp;1.3.6.1.4.1.21367.2005.3.7&amp;ISO"
   registryObject="theDocument">
   <rim:Name>
      <rim:LocalizedString value = "地域患者 ID (XDSDocumentEntry.patientId)"/>
   </rim:Name>
</rim:ExternalIdentifier>
```

図４　DocumentEntry - patientId（地域患者 ID）のメタデータ例

２．４　sourcePatientId（施設患者 ID）

ドキュメントエントリに関する施設患者 ID を指定する。

表６　DocumentEntry - sourcePatientId（施設患者 ID）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ExtrinsicObject/			
Slot				1..1
@name	スロットの名称。"**sourcePatientId**"に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	ドキュメントエントリに関する施設患者 ID をセット。	CX	R	1..1

```
<rim:Slot name="sourcePatientId">
   <rim:ValueList>
      <rim:Value>j98789^^^id.domain</rim:Value>
   </rim:ValueList>
</rim:Slot>
```

図５　DocumentEntry - sourcePatientId（施設患者 ID）のメタデータ例

２．５　sourcePatientInfo（患者基本情報）

ドキュメントエントリに関して、患者の基本情報を HL7V2.5 形式で指定する。

表７　DocumentEntry - sourcePatientInfo（患者基本情報）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ExtrinsicObject/			
Slot				1..1
@name	スロットの名称。"**sourcePatientInfo**"に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	ドキュメントエントリに関する患者情報をセット。 PID-3 は必須であり、ソース患者識別子を含む。 PID-5 は必須であり、患者名を含む。 PID-8 は必須であり、以下の患者の性別コードを含む。 M – 男性　F – 女性 O – その他 U – 不明 PID-7 は既知の場合は必須であり、患者の生年月日を含む。 PID-11 は既知の場合は必須であり、患者の住所を含む。 PID-2、PID-4、PID-12、PID-19 は使用してはならない。 その他の PID セグメントはオプションである。	PID	R	1..*

```
<rim:Slot name="sourcePatientInfo">
  <rim:ValueList>
    <rim:Value>PID-3|DTP-1^^^&amp;amp;1.3.6.1.4.1.21367.2005.3.7&amp;amp;ISO</rim:Value>
    <rim:Value>PID-5|DICTAPHONE^ONE^^^</rim:Value>
    <rim:Value>PID-7|19650120</rim:Value>
    <rim:Value>PID-8|M</rim:Value>
    <rim:Value>PID-11|100 MainSt^^BURLINGTON^MA^01803^USA</rim:Value>
  </rim:ValueList>
</rim:Slot>
```

図６　DocumentEntry - sourcePatientInfo（患者基本情報）のメタデータ例

２．６　classCode（文書クラス）

ドキュメントエントリに関して、ドキュメントの種類を指定する文書クラスを指定する。

表８　DocumentEntry - classCode（文書クラス）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ExtrinsicObject/			
Classification	分類識別子として指定。			0..*
@id	自身のオブジェクトを指す UUID を指定。	UUID	R	1..1
@classificationScheme	分類識別子で documentEntry.classCode を指す UUID。 "urn:uuid:41a5887f-8865-4c09-adf7-e362475b143a"固定。	UUID	固定	1..1
@classifiedObject	DocumentEntry（ExtrinsicObject）に割り当てられた UUID を指定。	UUID		1..1
@nodeRepresentation	分類ノードの表現形式を指定。 別途定める語彙 **A-classCode**(7.2.1)から選択したコード値。	classCode (基準A)	R	1..1
/Classification /Name				1..1
LocalizedString	コード値を日本語で表記。			1..1
@value	分類識別子の名称を指定。 別途定める語彙 **A-classCode** から選択したコード値の表示名。	classCode (基準A)	R	1..1
/Classification/Slot				1..1
@name	スロットの名称。"**codingScheme**"に固定。	char	固定	
/Slot/ValueList				1..1
/Slot/ValueList/Value	コード体系の表示名を指定。 **A-classCode** に固定。	char	固定	1..1

```
<rim:Classification
  id="c1"
  classificationScheme= "urn:uuid:41a5887f-8865-4c09-adf7-e362475b143a"
  classifiedObject="theDocument" nodeRepresentation="文書クラス" >
  <rim:Name>
    <rim:LocalizedString value="文書クラス表示名"/>
  </rim:Name>
  <rim:Slot name="codingScheme">
    <rim:ValueList>
      <rim:Value>A-classCode</rim:Value>
    </rim:ValueList>
  </rim:Slot>
</rim:Classification>
```

図７　DocumentEntry - classCode（文書クラス）のメタデータ例

２．７　typeCode（文書タイプ）

ドキュメントエントリに関して、ドキュメントの種類を指定する詳細な文書タイプを指定する。

表９　DocumentEntry - typeCode（文書タイプ）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ExtrinsicObject/			
Classification	分類識別子として指定。			0..*
@id	自身のオブジェクトを指す UUID を指定。	UUID	R	1..1
@classificationScheme	分類識別子で documentEntry.typeCode を指す UUID。 **"urn:uuid:41a5887f-8865-4c09-adf7-e362475b143a"** 固定。	UUID	固定	1..1
@classifiedObject	DocumentEntry（ExtrinsicObject）に割り当てられた UUID を指定。	UUID		1..1
@nodeRepresentation	分類ノードの表現形式を指定。 別途定める語彙 **B-typeCode**(7.2.2)から選択したコード値。	typeCode (基準 B)	R	1..1
/Classification /Name				1..1
LocalizedString	コード値を日本語で表記。			1..1
@value	分類識別子の名称を指定。 別途定める語彙 **B-typeCode** から選択したコード値の表示名。	typeCode (基準 B)	R	1..1
/Classification/Slot				1..1
@name	スロットの名称。"**codingScheme**"に固定。	char	固定	
/Slot/ValueList				1..1
/Slot/ValueList/Value	コード体系の表示名を指定。 **B-typeCode** に固定。	char	固定	1..1

```
<rim:Classification
  id="c2"
  classificationScheme= "urn:uuid:f0306f51-975f-434e-a61c-c59651d33983"
  classifiedObject="theDocument" nodeRepresentation="文書タイプ" >
  <rim:Name>
    <rim:LocalizedString value="文書タイプ表示名"/>
  </rim:Name>
  <rim:Slot name="codingScheme">
    <rim:ValueList>
      <rim:Value>B-typeCode</rim:Value>
    </rim:ValueList>
  </rim:Slot>
</rim:Classification>
```

図８　DocumentEntry - typeCode（文書タイプ）のメタデータ例

２．８　eventCodeList（イベントコード）

ドキュメントエントリに関して、ドキュメント化された診療行為をコードで指定する。

表１０　DocumentEntry - eventCodeList（イベントコード）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ExtrinsicObject/			
Classification	分類識別子として指定。			0..*
@id	自身のオブジェクトを指す UUID を指定。	UUID	R	1..1

@classificationScheme	分類識別子で documentEntry.eventCodeList を指す UUID。 "**urn:uuid:2c6b8cb7-8b2a-4051-b291-b1ae6a575ef4**"固定。	UUID	固定	1..1
@classifiedObject	DocumentEntry（ExtrinsicObject）に割り当てられた UUID を指定。	UUID		1..1
@nodeRepresentation	分類ノードの表現形式を指定。 別途定める語彙 **B-eventCode**(7.2.2)から選択したコード値。	eventCode (基準 B)	R	1..1
/Classification /Name				1..1
LocalizedString	コード値を日本語で表記。			1..1
@value	分類識別子の名称を指定。 別途定める語彙 **B-eventCode** から選択したコード値の表示名。	eventCode (基準 B)	R	1..1
/Classification/Slot				1..1
@name	スロットの名称。"**codingScheme**"に固定。	char	固定	
/Slot/ValueList				1..1
/Slot/ValueList/Value	コード体系の表示名を指定。 **B-eventCode** に固定。	char	固定	1..1

```
<rim:Classification
  id="c3"
  classificationScheme= "urn:uuid:2c6b8cb7-8b2a-4051-b291-b1ae6a575ef4"
  classifiedObject="theDocument" nodeRepresentation="イベント" >
  <rim:Name>
    <rim:LocalizedString value="イベント表示名"/>
  </rim:Name>
  <rim:Slot name="codingScheme">
    <rim:ValueList>
      <rim:Value>B-eventCode</rim:Value>
    </rim:ValueList>
  </rim:Slot>
</rim:Classification>
```

図９　DocumentEntry - eventCodeList（イベントコード）のメタデータ例

２．９　confidentialityCode（守秘レベル）

ドキュメントエントリに関して、ドキュメントエントリの守秘レベルを指定する。

表１１　DocumentEntry - confidentialityCode（守秘レベル）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	LeafRegistryObjectList/ExtrinsicObject/			
Classification	分類識別子として指定。			0..*
@id	自身のオブジェクトを指す UUID を指定。	UUID	R	1..1
@classificationScheme	分類識別子で documentEntry.confidentialityCodeList を指す UUID。 "**urn:uuid:f4f85eac-e6cb-4883-b524-f2705394840f**"固定。	UUID	固定	1..1
@classifiedObject	DocumentEntry（ExtrinsicObject）に割り当てられた UUID を指定。	UUID		1..1
@nodeRepresentation	分類ノードの表現形式を指定。 別途定める語彙 **A-confidentialityCode**(7.2.5)から選択したコード値。	confidentialityCode (基準 A)	R	1..1
/Classification /Name				1..1
LocalizedString	コード値を日本語で表記。			1..1

@value	分類識別子の名称を指定。 別途定める語彙 **A-confidentialityCode** から選択したコード値の表示名。	confidentialityCode (基準 A)	R	1..1
/Classification/Slot				1..1
@name	スロットの名称。"**codingScheme**"に固定。	char	固定	
/Slot/ValueList				1..1
/Slot/ValueList/Value	コード体系の表示名を指定。 **A-confidentialityCode** に固定。	char	固定	1..1

```
<rim:Classification
  classificationScheme= "urn:uuid:f4f85eac-e6cb-4883-b524-f2705394840f`
  classifiedObject="theDocument" nodeRepresentation="守秘レベル" >
  <rim:Name>
    <rim:LocalizedString value="守秘レベル表示名"/>
  </rim:Name>
  <rim:Slot name="codingScheme">
    <rim:ValueList>
      <rim:Value>A-confidentialityCode</rim:Value>
    </rim:ValueList>
  </rim:Slot>
</rim:Classification>
```

図１０　DocumentEntry - confidentialityCode（守秘レベル）のメタデータ例

２．１０　creationTime（作成日）

ドキュメントエントリに関して、ドキュメントの作成日時を(ドキュメントソース側で)記載する。

表１２　DocumentEntry - creationTime（作成日）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ExtrinsicObject/			
Slot				1..1
@name	スロットの名称。"**creationTime**"に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	ドキュメントを作成した日時をセットする。	DTM	R	1..1

```
<rim:Slot name="creationTime">
  <rim:ValueList>
  <rim:Value>20041225212010</rim:Value>
  </rim:ValueList>
</rim:Slot>
```

図１１　DocumentEntry - creationTime（作成日）のメタデータ例

２．１１　serviceStartTime（診療開始日）

ドキュメントエントリに関して、ドキュメントに関するサービスの開始日時を記載する。既知であれば、必須とする。

表１３　DocumentEntry - serviceStartTime（診療開始日）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ExtrinsicObject/			
Slot				1..1
@name	スロットの名称。"**serviceStartTime**"　に固定。	char	固定	
Slot/ValueList				1..1

Slot/ValueList/Value	サービスを開始した日時をセットする。	DTM	R	1..1

```
<rim:Slot name="serviceStartTime">
  <rim:ValueList>
    <rim:Value>20041225212010</rim:Value>
  </rim:ValueList>
</rim:Slot>
```

図１２　DocumentEntry - serviceStartTime（診療開始日）のメタデータ例

２．１２　serviceStopTime（診療終了日）

ドキュメントエントリに関して、ドキュメントに関するサービスの終了日時（退院日、転帰日など）を記載する。既知であれば、設定するものとする。条件 serviceStartTime ＜= serviceStopTime を満たすものとする。

表１４　DocumentEntry - serviceStopTime（診療終了日）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ExtrinsicObject/			
Slot				1..1
@name	スロットの名称。"**serviceStopTime**"に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	サービスを終了した日時をセットする。	DTM	O	1..1

```
<rim:Slot name="serviceStopTime">
  <rim:ValueList>
    <rim:Value>20041225232010</rim:Value>
  </rim:ValueList>
</rim:Slot>
```

図１３　DocumentEntry - serviceStopTime（診療終了日）のメタデータ例

２．１３　size（バイト長）

ドキュメントエントリに関して、ドキュメントのリポジトリに格納されるバイト長を記載する。サブミッションされたドキュメントのバイト長は、ドキュメントリポジトリが計算する。

表１５　DocumentEntry - size（バイト長）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ExtrinsicObject/			
Slot				1..1
@name	スロットの名称。"**size**"に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	ドキュメントエントリのオブジェクトのバイト値をセットする。	int	生成	1..1

```
<rim:Slot name="size">
  <rim:ValueList>
    <rim:Value>3654</rim:Value>
  </rim:ValueList>
</rim:Slot>
```

図１４　DocumentEntry - size（バイト長）のメタデータ例

２．１４　author（作成者）

ドキュメントエントリに関して、サブミッションするドキュメントの作成者に関する情報を記載する。以下の情報が含まれる。

(1) authorPerson（作成者）
ドキュメントエントリに関して、サブミッションするドキュメントの作成者を記載する。
(2) authorInstitution（作成者施設名称）
ドキュメントエントリに関して、作成者の所属する施設の名称を記載する。
(3) authorRole（作成者職種）
ドキュメントエントリに関して、ドキュメントの作成者の役割を記載する。
(4) authorSpecialty（作成者診療科）
ドキュメントエントリに関して、施設内の診療科を記載する。

表１６　DocumentEntry - author（作成者）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/RegistryPackage/			
Classification				0..*
@ classificationScheme	分類識別子で document.Author を指す UUID。 **"urn:uuid:93606bcf-9494-43ec-9b4ea7748d1a838d"**固定。	UUID	固定	
@ classifiedObject	DocumentEntry（ExtrinsicObject）に割り当てられた UUID を指定。	UUID	R	1..1
@ nodeRepresentation	分類ノードの表現形式を指定。ブランクに固定。	char	固定	1..1
Classification/Slot				1..1
@name	スロットの名称。**"authorPerson"**に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	作成者を記載。	XCN	R	1..1
Classification/Slot				0..*
@name	スロットの名称。**"authorInstitution"**に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	作者の所属する施設の名称を記載。	XON	O	1..1
Classification/Slot				0..*
@name	スロットの名称。**"authorRole"**に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	患者に対する作成者の役割を表現するコード。 別途定める語彙 **A-roleCode**(7.2.12)から選択したコード値。	RoleCode (基準 A)	O	1..*
Classification/Slot				0..*
@name	スロットの名称。**"authorSpecialty"**に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	サブミッションしたドキュメントの作成者の所属する施設内の診療科を記載。（診療科コード） 別途定める語彙 **B-practiceSettingCode**(7.2.8)から選択したコード値。	practiceSettingCodes (基準 B)	O	1..*

```
<rim:Classification
      classificationScheme="urn:uuid:93606bcf-9494-43ec-9b4ea7748d1a838d"
      classifiedObject="theDocument"
      nodeRepresentation="">
   <rim:Slot name="authorPerson">
      <rim:ValueList>
         <rim:Value>^東海^太郎^^^Dr^MD</rim:Value>
      </rim:ValueList>
   </rim:Slot>
   <rim:Slot name="authorInstitution">
      <rim:ValueList>
         <rim:Value>急性期 T 病院</rim:Value>
      </rim:ValueList>
   </rim:Slot>
   <rim:Slot name="authorRole">
      <rim:ValueList>
         <rim:Value>担当医</rim:Value>
      </rim:ValueList>
   </rim:Slot>
   <rim:Slot name="authorSpecialty">
      <rim:ValueList>
         <rim:Value>脳神経外科</rim:Value>
      </rim:ValueList>
   </rim:Slot>
</rim:Classification>
```

図１５　DocumentEntry - author（作成者）のメタデータ例

２．１５　comment（備考）

ドキュメントエントリに関して、コメントを記載する。

表１７　DocumentEntry - comment（備考）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/RegistryPackage/			
/Description/	サブミッションセットに関するコメントを記載。			0..1
LocalizedString	日本語で表記。			1..1
@value	連携パスでの使用目的などを別途定める。 自由形式のテキスト。	char	O	1..1

```
<rim:Description>
   <rim:LocalizedString value = "コメント"/>
</rim:Description>
```

図１６　DocumentEntry - comment（備考）のメタデータ例

２．１６　legalAuthenticator（作成元責任者）

ドキュメントエントリに関して、ドキュメントの法的認証者を記載する。

表１８　DocumentEntry - legalAuthenticator（作成元責任者）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/RegistryPackage/			
Slot				0..1

@name	スロットの名称。**"legalAuthenticator"** に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	作成者を記載。	XCN	O	1..1

```
<rim:Slot name="legalAuthenticator">
  <rim:ValueList>
    <rim:Value>^東海^太郎^^^Dr^MD</rim:Value>
  </rim:ValueList>
</rim:Slot>
```

図１７　DocumentEntry - legalAuthenticator（作成元責任者）のメタデータ例

２．１７　healthcareFacilityTypeCode（施設タイプ）

ドキュメントエントリに関して、ドキュメント化されている行為の実施期間中の診察機関、施設のタイプを記載する。

表１９　DocumentEntry - healthcareFacilityTypeCode（施設タイプ）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ExtrinsicObject/			
Classification	分類識別子として指定。			0..*
@id	自身のオブジェクトを指す UUID を指定。	UUID	R	1..1
@classificationScheme	分類識別子で documentEntry.healthcareFacilityTypeCode を指す UUID。 **"urn:uuid:f33fb8ac-18af-42cc-ae0e-ed0b0bdb91e1"**固定。	UUID	固定	1..1
@classifiedObject	DocumentEntry（ExtrinsicObject）に割り当てられた UUID を指定。	UUID	R	1..1
@nodeRepresentation	分類ノードの表現形式を指定。 別途定める語彙 **A-healthcareFacilityTypeCode** から選択したコード値。	HealthCareFacilityCode (基準 A)	R	1..1
/Classification /Name				1..1
LocalizedString	コード値を日本語で表記。			1..1
@value	分類識別子の名称を指定。 別途定める語彙 **A-healthcareFacilityTypeCode**(7.2.6)から選択したコード値の表示名。	HealthCareFacilityCode (基準 A)	R	1..1
/Classification/Slot				1..1
@name	スロットの名称。 **"codingScheme "**に固定。			
/Slot/ValueList				1..1
/Slot/ValueList/Value	コード体系の表示名を指定。 **A-healthcareFacilityTypeCode** に固定。	char	固定	1..1

```
<rim:Classification
  id="c4"
  classificationScheme= "urn:uuid:f33fb8ac-18af-42cc-ae0e-ed0b0bdb91e1"
  classifiedObject="theDocument" nodeRepresentation="施設タイプ" >
  <rim:Name>
    <rim:LocalizedString value="施設タイプ表示名"/>
  </rim:Name>
  <rim:Slot name="codingScheme">
    <rim:ValueList>
      <rim:Value>A-healthcareFacilityTypeCode </rim:Value>
    </rim:ValueList>
  </rim:Slot>
</rim:Classification>
```

図１８　DocumentEntry - healthcareFacilityTypeCode（施設タイプ）のメタデータ例

２．１８　practiceSettingCode（実施診療科）

ドキュメントエントリに関して、ドキュメント中で帰結した行為が実施された診療科等（clinical specialty）を記載する（例えば、係りつけ医、検査ラボ、放射線科など）。

表２０　DocumentEntry - practiceSettingCode（実施診療科）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ ExtrinsicObject/			
Classification	分類識別子として指定。			0..*
@id	自身のオブジェクトを指す UUID を指定。	UUID	R	1..1
@classificationScheme	分類識別子で documentEntry.practiceSettingCode を指す UUID。 **"urn:uuid:cccf5598-8b07-4b77-a05e-ae952c785ead"**固定。	UUID	固定	1..1
@classifiedObject	DocumentEntry（ExtrinsicObject）に割り当てられた UUID を指定。	UUID		1..1
@nodeRepresentation	分類ノードの表現形式を指定。 別途定める語彙 **B-practiceSettingCode**(7.2.8)から選択したコード値。	practice Setting Code (基準 B)	R	1..1
/Classification /Name				1..1
LocalizedString	コード値を日本語で表記。			1..1
@value	分類識別子の名称を指定。 別途定める語彙 **B-practiceSettingCode** から選択したコード値の表示名。	practice Setting Code (基準 B)	R	1..1
/Classification/Slot				1..1
@name	スロットの名称。**"codingScheme"**に固定。	char	固定	
/Slot/ValueList				1..1
/Slot/ValueList/Value	コード体系の表示名を指定。 **B-practiceSettingCode** に固定。	char	固定	1..1

```
<rim:Classification
  id="c5"
  classificationScheme="urn:uuid:cccf5598-8b07-4b77-a05e-ae952c785ead"
  classifiedObject="theDocument" nodeRepresentation="診療科" >
  <rim:Name>
    <rim:LocalizedString value="診療科表示名" />
  </rim:Name>
  <rim:Slot name="codingScheme">
    <rim:ValueList>
      <rim:Value> A-practiceSettingCode </rim:Value> </rim:ValueList>
  </rim:Slot>
</rim:Classification>
```

図１９　DocumentEntry - practiceSettingCode（実施診療科）のメタデータ例

２．１９　languageCode（記述言語）

ドキュメントエントリに関して、使用する記述言語を記載する。

表２１　DocumentEntry - languageCode（記述言語）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ExtrinsicObject/			
Slot				0..1
@name	スロットの名称。**"languageCode"**に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	言語は、日本語 **ja-JP** に固定。	char	固定	1..1

```
<rim:Slot name="languageCode">
  <rim:ValueList>
    <rim:Value>ja-JP</rim:Value>
  </rim:ValueList>
</rim:Slot>
```

図２０　DocumentEntry - languageCode（記述言語）のメタデータ例

２．２０　formatCode（フォーマットコード）

ドキュメントエントリに関して、ドキュメント形式を記載する。

表２２　DocumentEntry - formatCode（フォーマットコード）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ExtrinsicObject/			
Classification	分類識別子として指定。			0..*
@id	自身のオブジェクトを指す UUID を指定。	UUID	R	1..1
@classificationScheme	分類識別子で documentEntry.formatCode を指す UUID。 **"urn:uuid:a09d5840-386c-46f2-b5ad-9c3699a4309d"**固定。	UUID	固定	1..1
@classifiedObject	DocumentEntry（ExtrinsicObject）に割り当てられた UUID を指定。	UUID		1..1
@nodeRepresentation	分類ノードの表現形式を指定。 別途定める語彙 **A-formatCode**(7.2.9)から選択したコード値。	formatCode (基準 A)	R	1..1
/Classification /Name				1..1
LocalizedString	コード値を日本語で表記。			1..1

@value	分類識別子の名称を指定。 別途定める語彙 **A-formatCode** から選択したコード値の表示名。	formatCode (基準 A)	R	1..1
/Classification/Slot				1..1
@name	スロットの名称。**"codingScheme"**に固定。	char	固定	
/Slot/ValueList				1..1
/Slot/ValueList/Value	コード体系の表示名を指定。 **A-formatCode** に固定。	char	固定	1..1

```
<rim:Classification
  id="c6 "
  classificationScheme= "urn:uuid:a09d5840-386c-46f2-b5ad-9c3699a4309d"
  classifiedObject="theDocument" nodeRepresentation="フォーマット" >
  <rim:Name>
    <rim:LocalizedString value="フォーマット表示名"/>
  </rim:Name>
  <rim:Slot name="codingScheme">
    <rim:ValueList>
      <rim:Value>A-formatCode</rim:Value>
    </rim:ValueList>
  </rim:Slot>
</rim:Classification>
```

図２１　DocumentEntry - formatCode（フォーマットコード）のメタデータ例

２．２１　hash（ハッシュ値）

ドキュメントエントリに関して、ドキュメントのハッシュ値を（レポジトリで）計算しセットする。

表２３　DocumentEntry - hash（ハッシュ値）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ExtrinsicObject/			
Slot				1..1
@name	スロットの名称。**" hash"**に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	ハッシュ値（**SHA1** 値）をセットする。	int	生成	1..1

```
<rim:Slot name="hash">
  <rim:ValueList>
    <rim:Value>da39a3ee5e6b4b0d3255bfef95601890afd80709</rim:Value>
  </rim:ValueList>
</rim:Slot>
```

図２２　DocumentEntry - hash（ハッシュ値）のメタデータ例

２．２２　URI

ドキュメントエントリに関して、ドキュメントの取り出しに使用する URI をセットする。画像ファイルなど外部参照の際は、ドキュメントソースで設定、それ以外は、レポジトリで設定する。

表２４　DocumentEntry - URI のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/ExtrinsicObject/			
Slot				1..1
@name	スロットの名称。**" URI"**に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	**URI** 値をセットする。	URI	R	1..1

```
< rim:Slot name="URI">
  <rim:ValueList>
    <rim:Value>http://www.ihe.net</rim:Value>
  </rim:ValueList>
</rim:Slot>
```

２．２３　parentDocumentId 及び parentDocument Relationship

ドキュメントエントリに関して、既存の登録オブジェクトの UUID を指定する。CDA 文書（親）に添付する検査データ、画像データなどは、関係 APND で親子関係を指定する。

表２５　DocumentEntry - parentDocumentId 及び parentDocument Relationship のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/			
ObjectRef	登録オブジェクトの参照			0..*
@id	オブジェクトの識別子	UUID	R	
Association				0..*
@associationType	オブジェクトの関連を指定。 4 つの値（APND,RPLC,XFRM,signs）の内の 1 つ	UUID	R	1..1
@sourceObject	ソース側：登録するドキュメントエントリ	UUID	R	1..1
@targetObject	ターゲット側：親の既登録オブジェクト	UUID	R	1..1

```
<rim:ObjectRef id="urn:uuid:a6e06ca8-0c75-4064-9e5c-88b9045a96f6" />
<rim:Association
      associationType="APND の UUID"
      sourceObject="theDocument"
      targetObject="urn:uuid:a6e06ca8-0c75-4064-9e5c-88b9045a96f6" />
```

図２４　DocumentEntry - parentDocumentId 及び parentDocument Relationship のメタデータ例

３　サブミッションセット (SubmissionSet)

３．１　id, availabilityStatus, title

サブミッションセットを、ebRIM3.0 のレジストリパッケージ（RegistryPackage）として登録する。レジストリパッケージの状態（ライフサイクル）が、管理される。（XDS アダプタが値をセットする）

表２６　SubmissionSet - id, availabilityStatus, title のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/			
RegistryPackage	パッケージ(ebXML)			
@id	サブミッションセットの識別子（**id**）を指定。 （内部で UUID に変換される）。	char (UUID)	R	1..1
@status	・**availabilityStatus** を指定。 ・サブミッション処理後に、承認（Approved）状態にセットされる。 （ドキュメントソースからのサブミッション時には、指定不要）。 ・ドキュメントのオーナは、廃止（Deprecated）にできる。	UUID	R	1..1
/RegistryPackaget/Name				0..1
LocalizedString	日本語での表記。			1..1
@value	ドキュメントのタイトル（**title**）を記載。	char	O	1..1

```
<RegistryObjectList>

  <rim:RegistryPackage
    id="submissionSet"
    status="Approved の UUID" >
    <rim:Name>
      <rim:LocalizedString value="title"/>
    </rim:Name>
    ...
</rim:RegistryPackage>
```

図２５　SubmissionSet - id, availabilityStatus, title のメタデータ例

３．２　uniqueId（サブミッションセット ID）

サブミッションセットに関して、レジストリ内で一意に識別する ID（ドキュメントソース側で）指定する。

表２７　SubmissionSet - uniqueId（サブミッションセット ID）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/RegistryPackage/			
ExternalIdentifier	外部識別子として指定。			1..1
@id	自身のオブジェクトを指す UUID を指定。	UUID	R	1..1
@identificationScheme	外部識別子 uniqueId を指す UUID を指定。 **"urn:uuid:96fdda7c-d067-4183-912e-bf5ee74998a8"**に固定。	UUID	固定	1..1
@value	提出セットを識別する OID。	OID	R	1..1
@registryObject	SubmissionSet(RegistryPackage)に割り当てられた UUID を指定。	UUID	R	1..1
/ExternalIdentifier /Name				0..1
LocalizedString	日本語での表記。			1..1
@value	"**提出セット ID (XDSSubmissionSet.uniqueId)**"に固定	char	固定	1..1

```
<rim:ExternalIdentifier
   id="e3"
   identificationScheme= " urn:uuid:96fdda7c-d067-4183-912e-bf5ee74998a8"
   value="1.3.6.1.4.1.21367.2005.3.7.3670984664"
   registryObject="submissionSet">
   <rim:Name>
      rim:LocalizedString value = "提出セット ID(XDSSubmissionSet.uniqueId)"/>
   </rim:Name>
</rim:ExternalIdentifier>
```

図２６　SubmissionSet - uniqueId（サブミッションセット ID）のメタデータ例

３．３　patientID（地域患者 ID）

サブミッションセットに関して、地域患者 ID を指定する。

表２８　SubmissionSet - patientID（地域患者 ID）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/RegistryPackage/			
ExternalIdentifier	外部識別子として指定。			1..1
@id	自身のオブジェクトを指す UUID を指定。	UUID	R	1..1
@identificationScheme	外部識別子 uniqueId を指す UUID。 "urn:uuid:6b5aea1a-874d-4603-a4bc-96a0a7b38446"に固定。	UUID	固定	1..1
@value	PIX で発行された地域患者 ID を指定。 ・識別子発行機関 ID（PIX センタを指す OID） ・地域患者 ID（PIX で管理発行）	CX	R	1..1
@registryObject	SubmissionSet(RegistryPackage)に割り当てられた UUID を指定。	UUID	R	1..1
/ExternalIdentifier /Name				0..1
LocalizedString	日本語での表記。			1..1
@value	"地域患者 ID (XDSSubmissionSet.patientID)"に固定。	char	固定	1..1

```
<rim:ExternalIdentifier
   id="e4"
   identificationScheme= "urn:uuid:6b5aea1a-874d-4603-a4bc-96a0a7b38446"
   value="6578946^^^&amp;amp;1.3.6.1.4.1.21367.2005.3.7&amp;amp;ISO"
   registryObject="submissionSet">
   <rim:Name>
      <rim:LocalizedString value = "地域患者 ID (XDSSubmissionSet.patientID)"/>
   </rim:Name>
</rim:ExternalIdentifier>
```

図２７　SubmissionSet - patientID（地域患者 ID）のメタデータ例

３．４　sourceID（施設 ID）

サブミッションセットに関して、施設 ID を指定する。

表２９　SubmissionSet - sourceID（施設 ID）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/RegistryPackage/			
ExternalIdentifier	外部識別子として指定。			1..1

@id	自身のオブジェクトを指す UUID を指定。	UUID	R	1..1
@identificationScheme	外部識別子 uniqueId を指す UUID。 "**urn:uuid:554ac39e-e3fe-47fe-b233-965d2a147832**"に固定。	UUID	固定	1..1
@value	サブミッションした施設の施設 ID を指定。 ・識別子発行機関 ID（各施設を指す OID）	OID	R	1..1
@registryObject	SubmissionSet(RegistryPackage)に割り当てられた UUID を指定。	UUID	R	1..1
/ExternalIdentifier /Name				0..1
LocalizedString	日本語での表記。			1..1
@value	"**施設 ID (XDSSubmissionSet.sourceId)**"に固定	char	固定	1..1

```
<rim:ExternalIdentifier
   id="e5 "
   identificationScheme= "urn:uuid:554ac39e-e3fe-47fe-b233-965d2a147832"
   value="1.3.6.1.4.1.21367.2005.3.7"
   registryObject="submissionSet">
   <rim:Name>
      <rim:LocalizedString value = "施設 ID (XDSSubmissionSet.sourceId)"/>
   </rim:Name>
</rim:ExternalIdentifier>
```

図２８　SubmissionSet - sourceID（施設 ID）のメタデータ例

３．５　authorInstitution（作成者施設名称）

サブミッションセットに関して、作成者の所属する施設の名称を記載する。

表６－３０　SubmissionSet - authorInstitution（作成者施設名称）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/RegistryPackage/			
Slot				1..1
@name	スロットの名称。"**authorInstitution**"に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	作者の所属する施設の名称を指定。	XON	R	1..1

```
<rim:Slot name="authorInstitution">
   <rim:ValueList>
      <rim:Value>急性期 T 病院</rim:Value>
   </rim:ValueList>
</rim:Slot>
```

図２９　SubmissionSet - authorInstitution（作成者施設名称）のメタデータ例

３．６　submissionTime（提出日時）

サブミッションセットに関して、サブミッションした日時(ドキュメントソース側で指定)を記載する。

表３１　SubmissionSet - submissionTime（提出日時）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/RegistryPackage/			
Slot				1..1
@name	スロットの名称。"**submissionTime**"に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	サブミッションした時刻をセットする。	DTM	R	1..1

```
<rim:Slot name="submissionTime">
  <rim:ValueList>
  <rim:Value>20041225212010</rim:Value>
</rim:ValueList>
```

図３０　SubmissionSet - submissionTime（提出日時）のメタデータ例

３．７　author（作成者）

サブミッションセットに関して、作成者に関する情報を記載する。以下の情報が含まれる。

(1) authorPerson（作成者）

サブミッションセットに関して、サブミッションするドキュメントの作成者を記載する。

(2) authorInstitution（作成者施設名称）

サブミッションセットに関して、作成者の所属する施設の名称を記載する。

(3) authorRole（作成者職種）

サブミッションセットに関して、ドキュメントの作成者の役割を記載する。

(4) authorSpecialty（作成者診療科）

サブミッションセットに関して、施設内の診療科を記載する。

表３２　SubmissionSet - author（作成者）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/RegistryPackage/			
Classification				0..*
@ classificationScheme	分類識別子で document.Author を指す UUID。 **"urn:uuid:93606bcf-9494-43ec-9b4ea7748d1a838d"**固定。	UUID	固定	
@ classifiedObject	SubmissionSet（RegistryPackage）に割り当てられた UUID を指定。	UUID	R	1..1
@ nodeRepresentation	分類ノードの表現形式を指定。ブランクに固定。	char	固定	1..1
Classification/Slot				1..1
@name	スロットの名称。**"authorPerson"**に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	作成者を記載。	XCN	R	1..1
Classification/Slot				0..*
@name	スロットの名称。**"authorInstitution"**に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	作者の所属する施設の名称を記載。	XON	O	1..1
Classification/Slot				0..*
@name	スロットの名称。**"authorRole"**に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	患者に対する作成者の役割を表現するコード。 別途定める語彙**A-roleCode**(7.2.12)から選択したコード値。	RoleCode (基準 A)	O	1..*
Classification/Slot				0..*
@name	スロットの名称。**"authorSpecialty"**に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	サブミッションしたドキュメントの作成者の所属する施設内の診療科を記載。（診療科コード） 別途定める語彙 **B-practiceSettingCode**(7.2.8)から選択したコード値。	practiceSettingCodes (基準 B)	O	1..*

```
<rim:Classification
      classificationScheme="urn:uuid:93606bcf-9494-43ec-9b4ea7748d1a838d"
      classifiedObject="submissionSet "
      nodeRepresentation="">
   <rim:Slot name="authorPerson">
      <rim:ValueList>
         <rim:Value>^東海^太郎^^^Dr^MD</rim:Value>
      </rim:ValueList>
   </rim:Slot>
   <rim:Slot name="authorInstitution">
      <rim:ValueList>
         <rim:Value>急性期 T 病院</rim:Value>
      </rim:ValueList>
   </rim:Slot>
   <rim:Slot name="authorRole">
      <rim:ValueList>
         <rim:Value>担当医</rim:Value>
      </rim:ValueList>
   </rim:Slot>
   <rim:Slot name="authorSpecialty">
      <rim:ValueList>
         <rim:Value>脳神経外科</rim:Value>
      </rim:ValueList>
   </rim:Slot>
</rim:Classification>
```

図３１　SubmissionSet - author（作成者）のメタデータ例

４．８　comment（備考）

サブミッションセットに関して、コメントを記載する。

表３３　SubmissionSet - comment（備考）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/RegistryPackage/			
/Description/	サブミッションセットに関するコメントを記載。			0..1
LocalizedString	日本語で表記。			1..1
@value	連携パスでの使用目的などを別途定める。 自由形式のテキスト。	char	O	1..1

```
<rim:Description>
   <rim:LocalizedString value = "コメント"/>
</rim:Description>
```

図３２　SubmissionSet - comment（備考）のメタデータ例

４．９　contentTypeCode（内容タイプ）

サブミッションセットに関して、含まれる内容のタイプを指定する。

表３４　SubmissionSet - contentTypeCode（内容タイプ）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重

XPath	RegistryObjectList/RegistryPackage/			
Classification	分類識別子として指定。			0..*
@id	自身のオブジェクトを指す UUID を指定。	UUID	R	1..1
@classificationScheme	分類識別子で submissionSet.contentTypeCode を指す UUID。 **"urn:uuid:aa543740-bdda-424e-8c96-df4873be850"**固定。	UUID	固定	1..1
@classifiedObject	分類されるオブジェクトを指す。			1..1
@nodeRepresentation	分類ノードの表現形式を指定。 別途定める語彙 **A-classCode**(7.2.1)から選択したコード値。	classCode (基準A)	R	1..1
/Classification /Name				1..1
LocalizedString	コード値を日本語で表記。			1..1
@value	分類識別子の名称を指定。 別途定める語彙 **A-classCode** から選択したコード値の表示名。	classCode (基準A)	R	1..1
/Classification/Slot				1..1
@name	スロットの名称。**"codingScheme"**に固定。	char	固定	
/Slot/ValueList				1..1
/Slot/ValueList/Value	コード体系の表示名を指定。(codeDisplayNameList) **A-classCode** に固定。	char	固定	1..1

```
<rim:Classification
  id="c7 "
  classificationScheme= "urn:uuid:aa543740-bdda-424e-8c96-df4873be8500"
  classifiedObject="submissionSet" nodeRepresentation="内容タイプコード">
  <rim:Name>
    <rim:LocalizedString value="内容タイプコード表示名"/> </rim:Name>
  <rim:Slot name="codingScheme ">
  <rim:ValueList>
    <rim:Value>A-classCode</rim:Value>
  </rim:ValueList>
  </rim:Slot>
</rim:Classification>
```

図３３　SubmissionSet - contentTypeCode（内容タイプ）のメタデータ例

４　フォルダ（Folder）

４．１　id, availabilityStatus, title

フォルダを、ebRIM3.0 のレジストリパッケージ（RegistryPackage）として登録する。レジストリパッケージの状態（ライフサイクル）が、管理される。

表６－３５　Folder - id, availabilityStatus, title のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/			
RegistryPackage	パッケージ(ebXML)			
@id	フォルダの識別子（**id**）を指定。 （内部で UUID に変換される）。	UUID	R	1..1
@status	・**availabilityStatus** を指定。 ・サブミッション処理後に、承認（Approved）状態にセットされる。 （ドキュメントソースからのサブミッション時には、指定不要）。 ・ドキュメントのオーナは、廃止（Deprecated）にできる。	UUID	生成	1..1
/RegistryPackaget/Name				0..1

LocalizedString	日本語での表記。			1..1
@value	フォルダのタイトル（**title**）を記載。	char	R	1..1

```
< RegistryObjectList>

  <rim:RegistryPackage
    id="Folder"
    status="Approved の UUID" >
    <rim:Name>
      <rim:LocalizedString value="title"/>
    </rim:Name>
    . . .
  </rim:RegistryPackage>

  </RegistryObjectList>
```

図３４　Folder - id, availabilityStatus, title のメタデータ例

４．２　uniqueId（フォルダ識別番号）

フォルダに関して、レジストリ内で一意に識別する ID を(ドキュメントソース側で)指定する。

表３６　Folder - uniqueId（フォルダ識別番号）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/RegistryPackage/			
ExternalIdentifier	外部識別子として指定。			1..1
@id	自身のオブジェクトを指す UUID を指定。	UUID	R	1..1
@identificationScheme	外部識別子 uniqueId を指す UUID を指定。 "**urn:uuid:75df8f67-9973-4fbe-a900-df66cefecc5a**"に固定。	UUID	固定	1..1
@value	施設を識別する OID+施設内で発行する番号。	OID	R	1..1
@registryObject	Folder(RegistryPackage)に割り当てられた UUID を指定。	UUID	R	1..1

```
<rim:ExternalIdentifier
  id="e5 "
  identificationScheme= "urn:uuid:75df8f67-9973-4fbe-a900-df66cefecc5a "
  value="1.3.6.1.4.1.21367.2005.3.7.3670984664"
  registryObject="Folder"/>
```

図３５　Folder - uniqueId（フォルダ識別番号）のメタデータ例

４．３　patientID（地域患者 ID）

フォルダに関して、地域患者 ID を指定する。

表３７　Folder - patientID（地域患者 ID）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/RegistryPackage/			
ExternalIdentifier	外部識別子として指定。			1..1
@id	自身のオブジェクトを指す UUID を指定。	UUID	R	1..1
@identificationScheme	外部識別子 uniqueId を指す UUID。 "**urn:uuid:f64ffdf0-4b97-4e06-b79f-a52b38ec2f8a**"に固定。	UUID	固定	1..1

@value	PIX で発行された地域患者 ID を指定。 ・識別子発行機関 ID（PIX センタを指す OID） ・地域患者 ID（PIX で管理発行）	CX	R	1..1
@registryObject	Folder(RegistryPackage)に割り当てられた UUID を指定。	UUID	R	1..1
/ExternalIdentifier /Name				0..1
LocalizedString	日本語での表記。			1..1
@value	"**地域患者 ID (XDSFolder.patientID)**"に固定。	char	固定	1..1

```
<rim:ExternalIdentifier
   id="e6 "
   identificationScheme="urn:uuid:f64ffdf0-4b97-4e06-b79f-a52b38ec2f8a "
   value="6578946^^^&amp;1.3.6.1.4.1.21367.2005.3.7&amp;ISO"
   registryObject="Folder">
   <rim:Name>
      <rim:LocalizedString value = "地域患者 ID (XDSFolder.patientID)"/>
      </rim:Name>
</rim:ExternalIdentifier>
```

図６－３６　Folder - patientID（地域患者 ID）のメタデータ例

４．４　comment（備考）

フォルダに関して、コメントを記載する。

表３８　Folder - comment（備考）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/RegistryPackage/			
/Description/	フォルダに関するコメントを記載。			0..1
LocalizedString	日本語で表記。			1..1
@value	連携パスでの使用目的などを別途定める。 自由形式のテキスト。	char	O	1..1

```
<rim:Description>
   <rim:LocalizedString value = "コメント"/>
</rim:Description>
```

図３７　Folder - comment（備考）のメタデータ例

４．５　codeList（コードリスト）

フォルダに関して、格納する XDS ドキュメントのタイプ（診療行為を表現）を指定する。

表３９　Folder - codeList（コードリスト）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/RegistryPackage/			
Classification	分類識別子として指定。			0..*
@id	自身のオブジェクトを指す UUID を指定。	UUID	R	1..1
@classificationScheme	分類識別子で Folder.codeList を指す UUID。 "**urn:uuid:1ba97051-7806-41a8-a48b-8fce7af683c5**"に固定。	UUID	固定	1..1
@classifiedObject	Folder(RegistryPackage)に割り当てられた UUID を指定。	UUID	固定	1..1
@nodeRepresentation	分類ノードの表現形式を指定。 別途定める語彙 **B-typeCode**(7.2.2)から選択したコード値を指定。	TypeCode (基準 B)	R	1..1
/Classification /Name				1..1

LocalizedString	コード値を日本語で表記。			1..1
@value	分類識別子の名称を指定。 別途定める語彙 **B-typeCode** から選択したコード値の表示名を指定。	TypeCode (基準 B)	R	1..1
/Classification/Slot				1..1
@name	スロットの名称。"**codingScheme**" に固定。	char	固定	
/Slot/ValueList				1..1
/Slot/ValueList/Value	コード体系の表示名を指定。(codeDisplayNameList) **B-typeCode** に固定。	char	固定	1..1

```
<rim:Classification
  id="c8 "
  classificationScheme= 'urn:uuid:1ba97051-7806-41a8-a48b-8fce7af683c5'
  classifiedObject='Foloder' nodeRepresentation='codeList' >
  <rim:Name>
    <rim:LocalizedString value='コード値の表示名' />
  </rim:Name>
  <rim:Slot name='codingScheme'>
    <rim:ValueList>
      <rim:Value>B-typeCode</rim:Value>
    </rim:ValueList>
  </rim:Slot>
</rim:Classification>
```

図３８　Folder - codeList（コードリスト）のメタデータ例

４．６　lastUpdateTime（更新日付）

フォルダに関して、ドキュメントが登録され、フォルダに格納された時点のドキュメントレジストリでの時刻を記載する。この値は、ドキュメントレジストリがセットする。

表４０　Folder - lastUpdateTime（更新日付）のメタデータ定義

要素名/属性名	内容	型/語彙	設定	多重
XPath	RegistryObjectList/RegistryPackage/			
Slot				1..1
@name	スロットの名称。"**lastUpdateTime** " に固定。	char	固定	
Slot/ValueList				1..1
Slot/ValueList/Value	最新の更新時刻を、レジストリがセットする。	DTM	生成	1..1

```
<rim:Slot name="lastUpdateTime">
  <rim:ValueList>
    <rim:Value>20041225212010</rim:Value>
  </rim:ValueList>
</rim:Slot>
```

図４１　Folder - lastUpdateTime（更新日付）のメタデータ例

３．トランザクションの通信方式

XDS を構成するソース、リポジトリ、レジストリ及びコンシューマの各アクタ間のトランザクションでは、通信の方式として SOAP1.2 を採用している。さらに文書本体を取り扱う[ITI-41]と[ITI-43]では、SOAP1.2 に加え MTOM/XOP（MTOM with XOP encoding）形式を利用することが ITI-TF-2b において規定されている（3.41.5 及び 3.43.5 を参照）。各トランザクションの通信方式を図示したのが図 3-15 である。

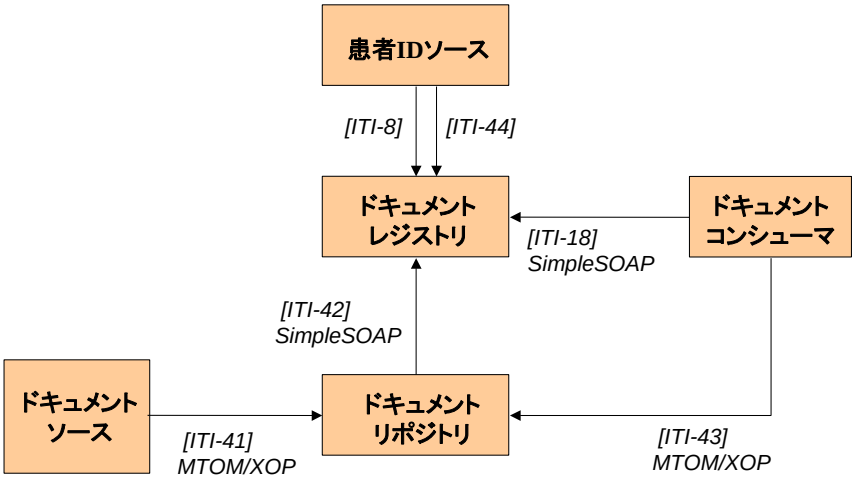

図 3-15　トランザクションの通信方式

以下、各トランザクションで取り扱われる SOAP メッセージの例を示す。

（1）文書・メタデータ登録[ITI-41]
ソースからリポジトリに対して出される文書登録要求の例を図 3-16 に、その要求例に対する返信メッセージを図 3-17 に示す。文書登録要求だけでなく、要求に対する返信メッセージも MTOM/XOP 形式を利用した SOAP メッセージであることに注意すること。
本例はソースから 1 つの文書を登録する場合の SOAP メッセージを表している。そのためメタデータとして、ドキュメントエントリ、サブミッションセット、アソシエーションが一つずつ SOAP メッセージの Body 部に入っている（ただし、図 3-16 では Slot、Classification、ExternalIdentifier で記述されるメタデータの属性値を省略している）。
また、SOAP メッセージの Body 部の下部に Document タグがあるが、これは文書本体とその文書のドキュメントエントリとの対応を記述する。文書本体は SOAP メッセージの添付データとして取り扱われる。
また、図 3-17 のメッセージは登録が成功した場合の返信メッセージである。これはメッセージ中の RegistryResponse タグに含まれる属性値 status が Success であることからわかる（図 3-17 の赤字部分）。もし、エラーが発生して登録が失敗した場合は、この属性値が Failure となり、SOAP の Body 部にエラーの内容が記載されている。

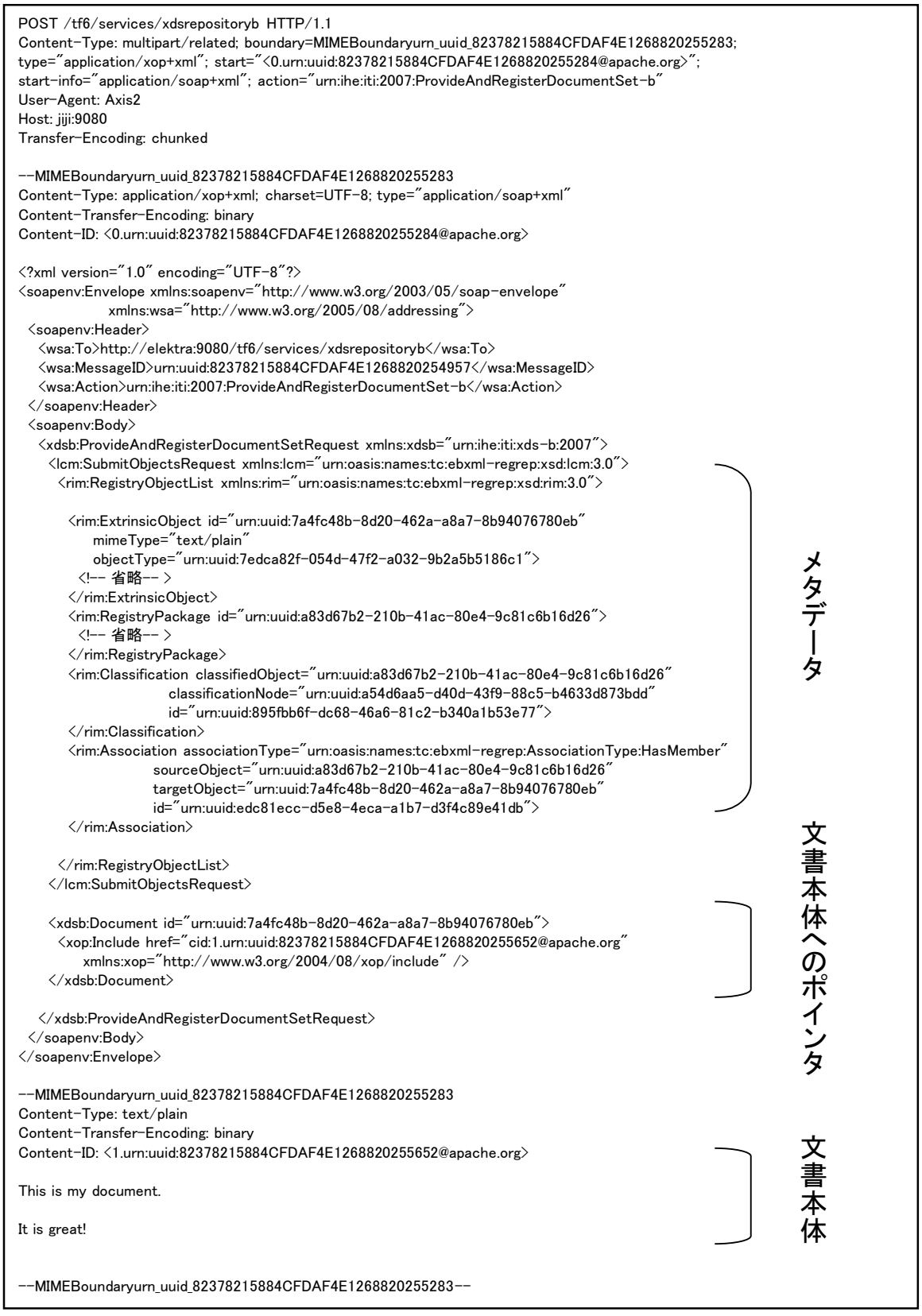

図 3-16 ドキュメントソースからの文書登録要求メッセージ

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: multipart/related; boundary=MIMEBoundaryurn_uuid_D93C36CAA0149E1BB01268820290558;
type="application/xop+xml"; start="0.urn:uuid:D93C36CAA0149E1BB01268820290559@apache.org";
start-info="application/soap+xml"; action="urn:ihe:iti:2007:ProvideAndRegisterDocumentSet-bResponse"
Transfer-Encoding: chunked
Date: Wed, 17 Mar 2010 10:04:50 GMT

--MIMEBoundaryurn_uuid_D93C36CAA0149E1BB01268820290558
Content-Type: application/xop+xml; charset=UTF-8; type="application/soap+xml"
Content-Transfer-Encoding: binary
Content-ID: <0.urn:uuid:D93C36CAA0149E1BB01268820290559@apache.org>

<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
                  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <soapenv:Header>
    <wsa:Action>urn:ihe:iti:2007:ProvideAndRegisterDocumentSet-bResponse</wsa:Action>
    <wsa:RelatesTo>urn:uuid:82378215884CFDAF4E1268820254957</wsa:RelatesTo>
  </soapenv:Header>
  <soapenv:Body>
    <rs:RegistryResponse  xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
                          status="urn:oasis:names:tc:ebxml-regrep:ResponseStatusType:Success" />
  </soapenv:Body>
</soapenv:Envelope>

--MIMEBoundaryurn_uuid_D93C36CAA0149E1BB01268820290558--
```

図 3-17 ドキュメントレジストリからの返信メッセージ

（2）メタデータの登録 [ITI-42]

リポジトリからレジストリに対して出されるメタデータ登録要求の例を図 3-18 に、その要求例に対する返信メッセージを図 3-19 に示す。メタデータの登録要求及びその返信メッセージは、MTOM/XOP を用いない通常の SOAP メッセージ（SIMPLE SOAP）を利用する。

本例は、リポジトリがソースからの 1 つの文書を登録する要求を受け取り、その中に含まれるメタデータを取り出して文書に関する追加情報（図 3-18 の中央部参照）を付与し、レジストリにこれらのメタデータを登録しようというものである。そのため SOAP メッセージの Body 部にドキュメントエントリ、サブミッションセット、アソシエーションが各１つ含まれている（ただし、図 3-18 ではメタデータの属性値を省略している）。

また、図 3-19 のメッセージは登録が成功した場合の返信メッセージである。これはメッセージ中の RegistryResponse タグに含まれる属性値 status が Success であることからわかる（図 3-19 の赤字部分）もし、エラーが発生して登録が失敗した場合は、この属性値が Failure となり、SOAP の Body 部にエラーの内容が記載されている。

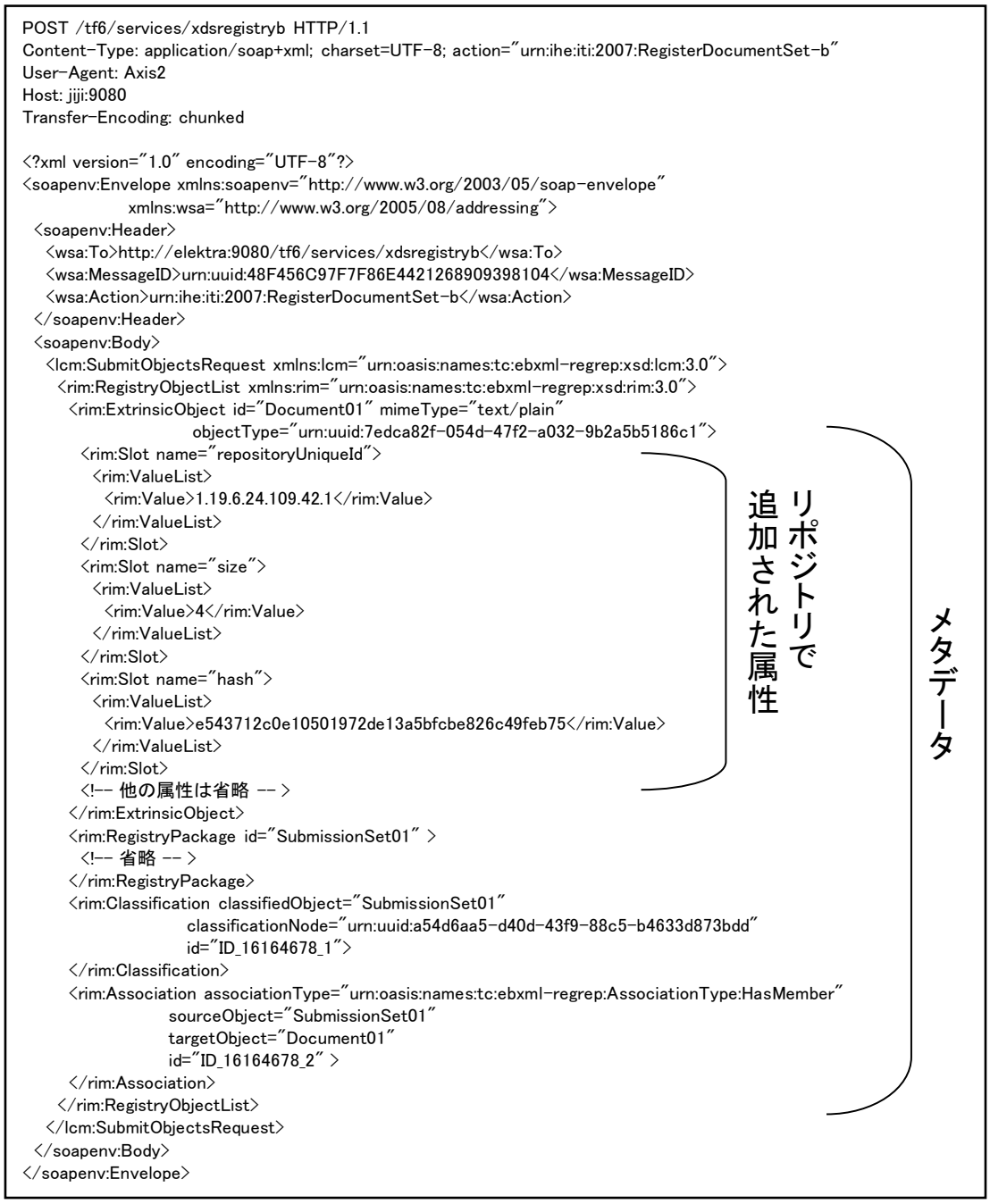

図 3-18 ドキュメントリポジトリからのメタデータ登録要求

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/soap+xml; action="urn:ihe:iti:2007:RegisterDocumentSet-bResponse";charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 18 Mar 2010 10:49:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
                  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <soapenv:Header>
    <wsa:Action>urn:ihe:iti:2007:RegisterDocumentSet-bResponse</wsa:Action>
    <wsa:RelatesTo>urn:uuid:48F456C97F7F86E4421268909398104</wsa:RelatesTo>
  </soapenv:Header>
  <soapenv:Body>
    <rs:RegistryResponse xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
                         status="urn:oasis:names:tc:ebxml-regrep:ResponseStatusType:Success" />
  </soapenv:Body>
</soapenv:Envelope>
```

図 3-19 ドキュメントレジストリからの返信メッセージ

（3）メタデータ検索要求 [ITI-18]

コンシューマからレジストリに対して出されるメタデータ検索要求メッセージの例を図 3-20 に、検索要求の例に対する結果（検索結果）のメッセージを図 3-21 にそれぞれ示す。検索要求及び検索結果は、MTOM/XOP を用いない通常の SOAP メッセージ（SIMPLE SOAP）を利用する。

先に触れたとおり、メタデータ検索要求はストアドクエリを利用する。本例では、ストアドクエリ「GetDocument」に基づくもので、検索条件は uniqueId が「160.27.80.47.54」あるいは「160.27.80.47.55」であるドキュメントエントリであることを意味する（図 3-20 中にある name が「$XDSDocumentEntryUniqueId」である Slot オブジェクトが検索条件を表す）。

図 3-21 に示す検索結果には、検索により見つかった、uniqueId が「160.27.80.47.54」であるドキュメントエントリと uniqueId が「160.27.80.47.55」であるドキュメントエントリの 2 つが含まれている（ただし、図 3-21 では uniqueId 以外のドキュメントエントリの属性値を省略している）。

```
POST /tf6/services/xdsregistryb HTTP/1.1
Content-Type: application/soap+xml; charset=UTF-8; action="urn:ihe:iti:2007:RegistryStoredQuery"
User-Agent: Axis2
Host: jiji:9080
Transfer-Encoding: chunked

<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
                  xmlns:wsa="http://www.w3.org/2005/08/addressing">
 <soapenv:Header>
  <wsa:To>http://elektra:9080/tf6/services/xdsregistryb</wsa:To>
  <wsa:MessageID>urn:uuid:6C0A7AC4D92351F45B1268908706961</wsa:MessageID>
  <wsa:Action>urn:ihe:iti:2007:RegistryStoredQuery</wsa:Action>
 </soapenv:Header>
 <soapenv:Body>
  <query:AdhocQueryRequest xmlns:query="urn:oasis:names:tc:ebxml-regrep:xsd:query:3.0"
                           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                           xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
                           xmlns="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0">
   <query:ResponseOption returnComposedObjects="true" returnType="LeafClass" />
   <AdhocQuery id="urn:uuid:5c4f972b-d56b-40ac-a5fc-c8ca9b40b9d4">
    <Slot name="$XDSDocumentEntryUniqueId">
     <ValueList>
      <Value>('160.27.80.47.1.54','160.27.80.47.1.55')</Value>
     </ValueList>
    </Slot>
   </AdhocQuery>
  </query:AdhocQueryRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

図 3-20 ドキュメントレジストリへの検索要求

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/soap+xml; action="urn:ihe:iti:2007:RegistryStoredQueryResponse";charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 18 Mar 2010 10:38:28 GMT

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
                  xmlns:wsa="http://www.w3.org/2005/08/addressing">
 <soapenv:Header>
  <wsa:Action>urn:ihe:iti:2007:RegistryStoredQueryResponse</wsa:Action>
  <wsa:RelatesTo>urn:uuid:6C0A7AC4D92351F45B1268908706961</wsa:RelatesTo>
 </soapenv:Header>
 <soapenv:Body>
  <query:AdhocQueryResponse xmlns:query="urn:oasis:names:tc:ebxml-regrep:xsd:query:3.0"
                            status="urn:oasis:names:tc:ebxml-regrep:ResponseStatusType:Success">
   <rim:RegistryObjectList xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0">

    <rim:ExtrinsicObject id="urn:uuid:5d57a8e7-fa67-4d08-bc88-d3fdd41036ba"
                         objectType="urn:uuid:7edca82f-054d-47f2-a032-9b2a5b5186c1"
                         status="urn:oasis:names:tc:ebxml-regrep:StatusType:Approved"
                         mimeType="text/plain" isOpaque="false" home=""
                         lid="urn:uuid:5d57a8e7-fa67-4d08-bc88-d3fdd41036ba">
     <!-- 省略 -- >
     <rim:ExternalIdentifier id="urn:uuid:5a7acd13-3ef7-46e7-8e80-a0af73fbe9d0"
              objectType="urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:ExternalIdentifier"
              identificationScheme="urn:uuid:2e82c1f6-a085-4c72-9da3-8640a32e42ab"
              value="160.27.80.47.1.54"
              home="" lid="urn:uuid:5a7acd13-3ef7-46e7-8e80-a0af73fbe9d0"
              registryObject="urn:uuid:5d57a8e7-fa67-4d08-bc88-d3fdd41036ba">
      <rim:Name>
       <rim:LocalizedString xml:lang="en-us" charset="UTF-8" value="XDSDocumentEntry.uniqueId" />
      </rim:Name>
      <rim:Description />
      <rim:VersionInfo versionName="1.1" />
     </rim:ExternalIdentifier>
    </rim:ExtrinsicObject>

    <rim:ExtrinsicObject id="urn:uuid:427a40e6-7214-425a-a020-ed83192b88ac"
                         objectType="urn:uuid:7edca82f-054d-47f2-a032-9b2a5b5186c1"
                         status="urn:oasis:names:tc:ebxml-regrep:StatusType:Approved"
                         mimeType="text/plain" isOpaque="false" home=""
                         lid="urn:uuid:427a40e6-7214-425a-a020-ed83192b88ac">
     <!-- 省略 -- >
     <rim:ExternalIdentifier id="urn:uuid:df573256-0d32-44bc-8954-be0123b5fba7"
              objectType="urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:ExternalIdentifier"
              identificationScheme="urn:uuid:2e82c1f6-a085-4c72-9da3-8640a32e42ab"
              value="160.27.80.47.1.55"
              home="" lid="urn:uuid:df573256-0d32-44bc-8954-be0123b5fba7"
              registryObject="urn:uuid:427a40e6-7214-425a-a020-ed83192b88ac">
      <rim:Name>
       <rim:LocalizedString xml:lang="en-us" charset="UTF-8" value="XDSDocumentEntry.uniqueId" />
      </rim:Name>
      <rim:Description />
      <rim:VersionInfo versionName="1.1" />
     </rim:ExternalIdentifier>
    </rim:ExtrinsicObject>

   </rim:RegistryObjectList>
  </query:AdhocQueryResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

図 3-21　レジストリからの検索結果

（4）文書の取得 [ITI-43]

コンシューマからリポジトリへ出される文書取得要求メッセージの例を図 3-22 に、このメッセージ例に対する結果として得られる文書取得結果メッセージの例を図 3-23 にそれぞれ示す。文書取得要求メッセージと文書取得結果メッセージは両方とも MTOM/XOP 形式を利用した SOAP メッセージであることに注意すること。文書取得要求では、取得したい文書があるリポジトリの uniqueId（repositoryUniqueId）と取得したい文書の uniqueId をメッセージ内で指定する（図 3-22 の下部にある DocumentRequest タグを参照）

```
POST /tf6/services/xdsrepositoryb HTTP/1.1
Content-Type: multipart/related;
boundary=MIMEBoundaryurn_uuid_6C0A7AC4D92351F45B1268908711218;
type="application/xop+xml";
start="<0.urn:uuid:6C0A7AC4D92351F45B1268908711219@apache.org>";
start-info="application/soap+xml"; action="urn:ihe:iti:2007:RetrieveDocumentSet"
User-Agent: Axis2
Host: jiji:9080
Transfer-Encoding: chunked

--MIMEBoundaryurn_uuid_6C0A7AC4D92351F45B1268908711218
Content-Type: application/xop+xml; charset=UTF-8; type="application/soap+xml"
Content-Transfer-Encoding: binary
Content-ID: <0.urn:uuid:6C0A7AC4D92351F45B1268908711219@apache.org>

<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
                  xmlns:wsa="http://www.w3.org/2005/08/addressing">
 <soapenv:Header>
  <wsa:To>http://elektra:9080/tf6/services/xdsrepositoryb</wsa:To>
  <wsa:MessageID>urn:uuid:6C0A7AC4D92351F45B1268908711215</wsa:MessageID>
  <wsa:Action>urn:ihe:iti:2007:RetrieveDocumentSet</wsa:Action>
 </soapenv:Header>
 <soapenv:Body>
  <RetrieveDocumentSetRequest xmlns="urn:ihe:iti:xds-b:2007"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="urn:ihe:iti:xds-b:2007
          file:/Users/bill/ihe/Frameworks/ITI-4/XDS.b/schema/IHE/XDS.b_DocumentRepository.xsd">

    <DocumentRequest>
     <RepositoryUniqueId>1.19.6.24.109.42.1.5</RepositoryUniqueId>
     <DocumentUniqueId>160.27.80.47.1.54</DocumentUniqueId>
    </DocumentRequest>

  </RetrieveDocumentSetRequest>
 </soapenv:Body>
</soapenv:Envelope>
--MIMEBoundaryurn_uuid_6C0A7AC4D92351F45B1268908711218--
```

要求するドキュメントの情報

図 3-22 リポジトリへの文書取得要求メッセージ

それに対して文書取得結果メッセージには、要求された文書本体が含まれる。図 3-23 の文書取得結果メッセージにおいて、SOAP メッセージの Body 部に DocumentResponse タグがあるが、これは取得した文書の情報が記述される。文書本体は SOAP メッセージの添付データとして取り扱われる。DocumentResponse タグと文書本体は、DocumentResponse タグ内にある Document タグで対応付けられる。

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: multipart/related;
boundary=MIMEBoundaryurn_uuid_D93C36CAA0149E1BB01268908713305;
type="application/xop+xml";
start="0.urn:uuid:D93C36CAA0149E1BB01268908713306@apache.org";
start-info="application/soap+xml"; action="urn:ihe:iti:2007:RetrieveDocumentSetResponse"
Transfer-Encoding: chunked
Date: Thu, 18 Mar 2010 10:38:32 GMT

--MIMEBoundaryurn_uuid_D93C36CAA0149E1BB01268908713305
Content-Type: application/xop+xml; charset=UTF-8; type="application/soap+xml"
Content-Transfer-Encoding: binary
Content-ID: <0.urn:uuid:D93C36CAA0149E1BB01268908713306@apache.org>

<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
                  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <soapenv:Header>
    <wsa:Action>urn:ihe:iti:2007:RetrieveDocumentSetResponse</wsa:Action>
    <wsa:RelatesTo>urn:uuid:6C0A7AC4D92351F45B1268908711215</wsa:RelatesTo>
  </soapenv:Header>
  <soapenv:Body>
    <xdsb:RetrieveDocumentSetResponse xmlns:xdsb="urn:ihe:iti:xds-b:2007">
      <rs:RegistryResponse xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
                           status="urn:oasis:names:tc:ebxml-regrep:ResponseStatusType:Success" />

      <xdsb:DocumentResponse>
        <xdsb:RepositoryUniqueId>1.19.6.24.109.42.1.5</xdsb:RepositoryUniqueId>
        <xdsb:DocumentUniqueId>160.27.80.47.1.54</xdsb:DocumentUniqueId>
        <xdsb:mimeType>text/plain</xdsb:mimeType>
        <xdsb:Document>
          <xop:Include href="cid:1.urn:uuid:D93C36CAA0149E1BB01268908713308@apache.org"
                       xmlns:xop="http://www.w3.org/2004/08/xop/include" />
        </xdsb:Document>
      </xdsb:DocumentResponse>

    </xdsb:RetrieveDocumentSetResponse>
  </soapenv:Body>
</soapenv:Envelope>

--MIMEBoundaryurn_uuid_D93C36CAA0149E1BB01268908713305
Content-Type: text/plain
Content-Transfer-Encoding: binary
Content-ID: <1.urn:uuid:D93C36CAA0149E1BB01268908713308@apache.org>

This is my document.

It is great!

--MIMEBoundaryurn_uuid_D93C36CAA0149E1BB01268908713305--
```

取得したドキュメントの情報

取得したドキュメント本体

図 3-23 リポジトリからの文書取得結果

4. *XDS* Metadata Validator

http://gazelle.ihe.net/content/xds-metadata-validator