Koordinierungsstelle
für IT-Standards

IT-Planungsrat

Spezifikation – XTA Erweiterung Bulk Fetching

Version 1 / Draft – 26. Oktober 2023

Der Standard XTA 2 wird im Auftrag des IT-Planungsrates von der KoSIT betrieben und innerhalb einer definierten Gremienstruktur weiterentwickelt. Die vorliegende Spezifikation und weitere Informationen sind auf den Seiten der KoSIT unter www.xoev.de zu finden.

Inhaltsverzeichnis

Einleitung

Diese Spezifikation wendet sich an Entwicklerinnen und Entwickler, welche die von XTA bereitgestellte Funktionalität in ihren Anwendungen implementieren.

Die vorliegende Erweiterung für den Standard XTA 2 ermöglicht einen sukzessiven Abruf von Nachrichten. Dazu werden neue Methoden (siehe Abschnitt Kapitel 2 auf Seite 3) ergänzend zum XTA Kern zur Verfügung gestellt.

1 Voraussetzungen

Die Erweiterung *Bulk Fetching* verwendet in der vorliegenden Version Datentypen aus XTA 2 Version 5 und kann somit nur der mit genannten Version des Kerns eingesetzt werden.

1.1 Konformität

Gegenstand der Konformitätsbewertung ist ein Software-Produkt, welches den Transportadapter für die Anwendungsebene oder für die Transportebene implementiert. Eine Implementierung für die Anwendungs-/Transportebene ist genau dann „konform zur Spezifikation *Bulk Fetching Version 1*“, wenn alle Vorgaben aus der Spezifikation an die umgesetzte Rolle erfüllt wurden. Die Konformitätserklärung der Produktherstellerin erfolgt freiwillig und muss mindestens über folgende Informationen verfügen:

- Eindeutige Kennzeichnung als Selbsterklärung zur XTA-Konformität
- Eindeutiger Verweis auf die XTA-Spezifikation mit Angabe von Version und Datum, die umgesetzt wurde
  - Verweis auf die Quelle der Spezifikation (Link zur offiziellen Seite)
  - Es kann stets nur auf die XTA-Spezifikation verwiesen werden, nicht auf die Hilfsmittel, die zusammen mit der XTA-Spezifikation veröffentlicht werden (z.B. einzelne Schema- oder WSDL-Dateien)
- Eindeutige Identifikation des Produktes, insbesondere die Produktbezeichnung einschließlich Versionsangabe
  - Angabe, welche Rollen umgesetzt worden sind
- Eindeutige Angabe der Herstellerin einschließlich gültiger Kontaktinformationen
- Datum, Ort und Unterschrift einer vertretenden Person des Herstellers bzw. der Herstellerin

2 Kommunikationsmodell

In diesem Kapitel werden die Schnittstellen für die Kommunikation zwischen der Anwendungsebene und Transportebene beschrieben. Hiermit wird auch festgelegt, welche Rolle welche Schnittstellen bereitstellen und aufrufen können muss.

Beim sukzessiven Abruf von Fachnachrichten aus dem Postfach muss die Ressourcenkennung `MessageBoxIteratorID` der zusätzlich parametrisierten Methode `close` des Kerns und der Methode `getNextMessage` übergeben werden. Die Ressourcenkennung wird durch den Aufruf der Methode `createIterator` erzeugt. Im nachfolgenden Prozess wird die Abholung von Fachnachrichten durch eine aktive Leserinstanz beschrieben:

1. **Erzeugung der Ressourcenkennung**

   Mit dem ersten Aufruf bekommt die Instanz des Lesers eine Ressourcenkennung für einen Iterator, den sie für die weitere Abholung von Fachnachrichten benötigt. Die dem Iterator zugewiesenen Fachnachrichten werden für anderweitige Abholungen gesperrt. Die maximale Dauer der Sperre wird unter den Kommunikationspartnern individuell abgestimmt. Wenn mehrere Instanzen des Lesers auf die selektierten Fachnachrichten zugreifen wollen, gibt die erste Instanz den Iterator an andere Leserinstanzen weiter.

   ☞ Methode `createIterator` mit Übergabe der Selektionskriterien und Rückgabe der Ressourcenkennung (siehe Abschnitt 2.2.1.1 auf Seite 4)

2. **Abholung der ersten Fachnachricht**

   Unter Angabe der Ressourcenkennung wird die erste Fachnachricht durch eine Leserinstanz abgeholt.

   ☞ Methode `getNextMessage` ohne Übergabe der `XTAMessageID` beim ersten Aufruf (siehe Abschnitt 2.2.1.2 auf Seite 5)

3. **Überprüfung der Kommunikation**

   Die Instanz des Lesers überprüft, ob der Transport der Fachnachricht nach allen Vorgaben des Serviceprofils erfolgreich durchgeführt werden konnte, z. B. ob die verwendeten Zertifikate gültig waren. Bei positivem Ergebnis kann sie die Fachnachricht des Autors verarbeiten. Im Falle eines Misserfolgs muss sie ggf. Eskalationsmaßnahmen ergreifen.

   ☞ Methode `getTransportReport` (siehe XTA Kern)

4. **Abholen weiterer Fachnachrichten**

   Solange mindestens eine weitere Fachnachricht zur Abholung vorliegt, wird diese von der Instanz des Lesers beim Empfänger mit der Ressourcenkennung abgerufen. Die `XTAMessageID` der zuletzt abgerufenen Fachnachricht wird zwecks Abholungsbestätigung mitübergeben (Funktionalität der Methode `close`). Der Status des Iterators wird über den Rückgabeparameter `HasNextMessage` zurückgegeben.

   ☞ Methode `getNextMessage` inklusive `XTAMessageID` zur Bestätigung der vorherigen Abholung (siehe Abschnitt 2.2.1.2 auf Seite 5)

5. <*Wiederholbare Prozessschritte* >

Die beiden vorhergehenden Arbeitsschritte können wiederholt werden bis alle gewünschten Fachnachrichten abgeholt wurden.

☞ Methoden `getNextMessage` (siehe Abschnitt 2.2.1.2 auf Seite 5) und `getTransportReport` (siehe XTA Kern)

6. **Beenden der Abholung von Nachrichten**

   Wenn die Instanz des Lesers alle gewünschten Fachnachrichten abgeholt hat, muss sie dies dem Empfänger mitteilen, indem sie abschließend eine Bestätigung für die zuletzt abgeholte Fachnachricht sendet. Diese Information unterstützt den Empfänger bei der Koordination des parallelen Zugriffs. Mit dem Beenden der sukzessiven Abholung von Fachnachrichten wird der Iterator und dadurch für die anderweitige Abholung nicht verfügbare Nachrichten (falls vorhanden) freigegeben.

   ☞ Methode `close` (siehe XTA Kern)

Für eine parallele Abholung durch mehrere Instanzen eines Lesers, reserviert die erste Instanz in Prozessschritt 1 eine Ressourcenkennung. Diese reicht sie an die andere Instanz weiter, die dann die Prozessschritte 2, 3 und 4 parallel zu ihr durchführt.

2.1 Anwendungsebene

Der Leser muss nachfolgende Methoden des Empfängers im **asynchronen** Kommunikationsszenario aufrufen können:

- createIterator
- getNextMessage

2.2 Transportebene

Der Empfänger muss dem Leser nachfolgende Methoden im **asynchronen** Kommunikationsszenario bereitstellen:

- createIterator
- getNextMessage

2.2.1 Schnittstellentyp extension.bulkFetching.recipient

In diesem Schnittstellentyp sind alle Methoden zusammengefasst, die vom Empfänger dem Leser im asynchronen Kommunikationsszenario zur Verfügung gestellt werden müssen.

2.2.1.1 Methode createIterator

Die Methode erzeugt eine Ressourcenkennung für einen Iterator und sperrt die von Selektionskriterien betroffenen Fachnachrichten für den Leser.

2.2.1.1.1 Exceptions

`NotImplementedException`

Die Exception signalisiert, dass diese Methode nicht implementiert wurde.

`ParameterNotSupportedException`

Die Exception signalisiert den Aufruf einer Methode mit spezifikationsgemäßen, aber von der Implementierung nicht unterstützen Parametern.

`PermissionDeniedException`

Die Exception signalisiert einen Autorisierungsfehler.

`TechnicalProblemException`

Die Exception signalisiert einen allgemeinen Fehler während der Verarbeitung.

2.2.1.1.2 Input

Tabelle 2.1. Header

WSDL Message	Message Part	Part Element
XTAHeader	Party	xta-core:Party (vgl. Seite 10)
Element des Typs PartyType mit Informationen, die zur Autorisierung von Methodenaufrufen verwendet wird. Es ergänzt damit die Informationen, die bereits aus dem Verbindungsaufbau (z.B. HTTP) vorhanden sind.		

Tabelle 2.2. Body

WSDL Message	Message Part	Part Element
CreateIteratorRequest	MessageSelector	xta-core:MessageSelector (vgl. Seite 10)
Hiermit wird die Auswahl- oder Filterung von Nachrichten aus einem Postfach vorgenommen.		

2.2.1.1.3 Output

Tabelle 2.3. Header

WSDL Message	Message Part	Part Element
XTAHeader	MessageBoxStatus	xta-core:MessageBoxStatus (vgl. Seite 10)
Diese Struktur enthält Zusatzinformationen zur Anfrage an das Postfachs unter Berücksichtigung der gesetzten Auswahl- oder Filterkritierien.		

Tabelle 2.4. Body

WSDL Message	Message Part	Part Element
CreateIteratorResponse	MessageBoxIteratorID	bulkFetching:MessageBoxIteratorID (vgl. Seite 7)
Die Ressourcenkennung (Iterator) für die sukzessive Abholung von Fachnachrichten aus dem Postfach.		

2.2.1.2 Methode getNextMessage

Die Methode ermöglicht das sukzessive Abholen von Fachnachrichten unter Angabe der Ressourcenkennung des Iterators. Es können auch parallele Abholungen durchgeführt werden.

2.2.1.2.1 Exceptions

`NotImplementedException`

Die Exception signalisiert, dass diese Methode nicht implementiert wurde.

`ParameterNotSupportedException`

Die Exception signalisiert den Aufruf einer Methode mit spezifikationsgemäßen, aber von der Implementierung nicht unterstützen Parametern.

`PermissionDeniedException`

Die Exception signalisiert einen Autorisierungsfehler.

`TechnicalProblemException`

Die Exception signalisiert einen allgemeinen Fehler während der Verarbeitung.

2.2.1.2.2 Input

Tabelle 2.5. Header

WSDL Message	Message Part	Part Element
XTAHeader	Party	xta-core:Party (vgl. Seite 10)
Element des Typs PartyType mit Informationen, die zur Autorisierung von Methodenaufrufen verwendet wird. Es ergänzt damit die Informationen, die bereits aus dem Verbindungsaufbau (z.B. HTTP) vorhanden sind.		

Tabelle 2.6. Body

WSDL Message	Message Part	Part Element
GetNextMessageRequest	MessageFetchIterator	bulkFetching:MessageFetchIterator (vgl. Seite 7)
Hiermit müssen die Steuerungsdaten für den sukzessiven Nachrichtenabruf übertragen werden.		

2.2.1.2.3 Output

Tabelle 2.7. Header

WSDL Message	Message Part	Part Element
BulkFetchingHeader	HasNextMessage	bulkFetching:HasNextMessage (vgl. Seite 8)
Diese Struktur enthält den Status des Iterators.		
XTAHeader	MessageMetaData	xta-core:MessageMetaData (vgl. Seite 10)
In dieser Struktur werden die Steuerungsdaten des Transportauftrags festgelegt.		

Tabelle 2.8. Body

WSDL Message	Message Part	Part Element
GetNextMessageResponse	ContentContainer	xta-core:ContentContainer (vgl. Seite 10)
Hiermit muss die nächste, noch nicht abgeholte Fachnachricht zu einer vorausgegangenen Abfrage übertragen werden. Hierbei ist zu beachten, dass für eine abgeholte Fachnachricht der Zeitstempel `Reception` erst gesetzt wird, nachdem der Abholauftrag durch den Leser entweder mit der Methode `close` oder der Methode `getNextMessage` bestätigt worden ist.		

3 Datenmodell

In diesem Kapitel werden die innerhalb der Schnittstellen verwendeten Datentypen für die Kommunikation zwischen der Anwednungsebene und Transportebene beschrieben.

3.1 Zusätzliche Parametrisierung des XTA 2 Kerns

Die Datentypen der nachfolgenden Unterkapitel werden über die XTA 2 Kernparameter gemäß der Spezifikation XTA 2 Version 5 zusätzlich übertragen.

3.1.1 MessageBoxCloseRequestExtensionType

Typ: *MessageBoxCloseRequestExtensionType*

Kindelement von *MessageBoxCloseRequestExtensionType*				
Kindelement	**Typ**	**Anz.**	**Ref.**	**Seite**
MessageBoxIteratorID	*xs:anyURI*	**1**		
Die Ressourcenkennung (Iterator) des vorangegangenen Methodenaufrufs zur Zuordnung zum ursprünglichen Abholauftrag.				

3.2 MessageBoxIteratorID

Globales Element: *MessageBoxIteratorID*

Element des Typs MessageBoxIteratorIDType (siehe Abschnitt 3.3 auf Seite 7).

3.3 MessageBoxIteratorIDType

Typ: *MessageBoxIteratorIDType*

Ressourcenkennung (Iterator) für die sukzessive Abholung von Fachnachrichten.

Dieser Typ ist eine Einschränkung des Basistyps *xs:anyURI*.

3.4 MessageFetchIterator

Globales Element: *MessageFetchIterator*

Element des Typs MessageFetchIteratorType (siehe Abschnitt 3.5 auf Seite 7).

3.5 MessageFetchIteratorType

Typ: *MessageFetchIteratorType*

Struktur zur Bestätigung des Erhalts und Steuerung der sukzessiven, parallelen Abholung von Nachrichten.

Kindelemente von *MessageFetchIteratorType*				
Kindelement	**Typ**	**Anz.**	**Ref.**	**Seite**
MessageBoxIteratorID	*xs:anyURI*	**1**		
Die Ressourcenkennung (Iterator) des vorangegangenen Methodenaufrufs zur Zuordnung zum ursprünglichen Abholauftrag. Die ID kann an andere Leser weitergereicht werden, falls mehrere Leser auf die Nachrichten zugreifen wollen.				
XTAMessageID	*xta-core:XTAMessageIDType*	**0..1**	A.1	10
XTAMessageID der letzten, vom Leser erhaltenen Nachricht als Bestätigung über den Erhalt. Sobald der erfolgreiche Abruf durch den Leser signalisiert wird, setzt der Empfänger den Zeitstempel „Reception“ in MessageMetaData der Nachricht auf seine Systemzeit.				

3.6 HasNextMessage

Globales Element: *HasNextMessage*

Struktur zur Angabe des Status des Iterators

Kindelement von *HasNextMessage*				
Kindelement	**Typ**	**Anz.**	**Ref.**	**Seite**
HasNextMessage	*xs:boolean*	**1**		
Gibt an, ob mindestens eine weitere Fachnachricht zur Abholung vorliegt.				

4 Fehlerbehandlung

Es werden nur Exceptions aus dem XTA Kern wiederverwendet (siehe Anhang Externe Modelle).

A Eingebundene externe Modelle/ Standards

Folgende externe Modelle werden in dieser Spezifikation verwendet und sind auf den XÖV-Webseiten (siehe http://www.xoev.de/de/produkte) oder im XRepository (siehe http://www.xrepository.de) veröffentlicht:

A.1 XTA 2 Version 5

XTA 2; Version 5

Folgende Datentypen aus dem externen Modell werden in dieser Spezifikation verwendet:

- ContentContainer
- InvalidMessageIDException
- MessageBoxStatus
- MessageMetaData
- MessageSchemaViolationException
- MessageSelector
- MessageVirusDetectionException
- NotImplementedException
- ParameterIsNotValidException
- ParameterNotSupportedException
- Party
- PermissionDeniedException
- SyncAsyncException
- TechnicalProblemException
- XTAMessageIDType

B Versionshistorie

B.1 Release E, Version 1 / Draft (26. Oktober 2023)

CR 2021-09 Erweiterung Bulk Fetching

Im Rahmen der Aufteilung von XTA 2 in einen Kern und die Erweiterungen wurden die Inhalte dieser Spezifikation als eine Erweiterung definiert.

Die Methode `getNextMessage` wurde überarbeitet und erfordert die Nutzung der neuen Methode `createIterator`. Der neue Parameter `hasNextMessage` wird anstatt `MsgBoxResponse` genutzt um das Vorhandensein weiterer Nachrichten anzuzeigen.

CR 2022-05 Naming bei MessageID und MsgBoxRequestID

Das Element `MsgBoxRequestID` wurde zu `MessageBoxIteratorID` umbenannt und die Beschreibung zwecks besserer Nachvollziehbarkeit geändert.

Das Element `LastMsgReceived` wurde zu `XTAMessageID` umbenannt um die Verwechslungsgefahr mit anderen Identifikatoren auszuschließen.