

ITML
DISTRIBUTED SESSION
MANAGEMENT
SPECIFICATION
WORKING DRAFT



Last Updated: 01/30/01

Control Number:

Version: 0.5

Editor(s):
David Orchard
David Tarrell
Gilbert Pilz

Copyright (C) Jamcracker, Inc.
All Rights Reserved

Jamcracker, Inc.

ITML VISION	1
ABSTRACT	1
STATUS OF THIS DOCUMENT	1
RELATIONSHIP TO OTHER STANDARDS.....	1
<i>Normative</i>	1
AUDIENCE.....	2
DOCUMENT CONVENTIONS.....	2
USE CASES AND REQUIREMENTS	3
1.A. BROWSER-BASED SINGLE SIGN-ON TO ECOSYSTEM	3
1.B. LOG-OFF.....	3
1.C. TIME-OUT.....	3
1.D. TRAVERSAL TO MULTIPLE PARTNER SITES	3
REQUIREMENTS.....	3
ASSUMPTIONS.....	3
TERMINOLOGY	4
SESSION MANAGEMENT ARCHITECTURE	5
DEPLOYMENT DIAGRAM.....	5
SESSION MANAGEMENT INTERACTION DIAGRAMS	5
ITML SESSION MANAGEMENT CLASS DIAGRAM.....	7
DESIGN GUIDELINES.....	8
SESSION MANAGEMENT MESSAGES	9
NAMESPACE.....	9
REQUESTS.....	9
ERRORS.....	9
SAMPLES.....	9
SAMPLE REQUEST - GET SESSION	9
SAMPLE RESPONSE.....	10
SAMPLE RESPONSE, ERROR	10
APPENDIX.....	11
SESSION MESSAGE SCHEMAS.....	11
SESSION STRUCTURE SCHEMAS.....	11
FUTURES.....	12
IMPORTANT DESIGN NOTES (NON-NORMATIVE).....	12
<i>Error Handling and Recovery</i>	12
CREDITS.....	13
REVISION HISTORY	13

ITML Vision

ITML - the Information Technology Markup Language - is a set of specifications of protocols, message formats and best practices in the ASP and ASP aggregation market to provide seamless integration of partners and business processes. It is based on open standards, particularly XML and HTTP. It also uses emerging standards, particularly SOAP and XML Schema.

Abstract

This document provides a framework for specific interactions around session management to occur between Jamcracker and a partner. A typical use of this is for the synchronization of a single sign-on assertion. It addresses the needs of single sign-on, single sign-off, single time-out, and single maintain session.

This specification describes the following key decisions:

- ITML Message and Protocol is the transmission format
- Jamcracker pulls state changes from ASPs
- Jamcracker can pull from many.
- Times of actions on state changes are deltas from last pull, not absolute
- The entire session for a given user is sent for each request

Status of this Document

This document is a Working Draft, issued by the Jamcracker ITML team, for review by selected partners.

The ITML team expects that significant changes will occur in this document before version 1.0 is released. The ITML Team will not allow early implementation to constrain its ability to make changes to this specification prior to final release.

Relationship to other standards

The ITML Framework 1.0 has been influenced by many recent standards efforts including, but not limited to, the following:

Normative

XML:

<http://www.w3.org/TR/REC-xml>

SOAP:

<http://www.w3.org/TR/SOAP/>

XML Schema

ITML Messaging and Protocol

[http://www.itml.org/Specifications/ITML Messaging And Protocol2000-11-22.doc](http://www.itml.org/Specifications/ITML_Messaging_And_Protocol2000-11-22.doc)

Audience

This document is a technical specification and is intended for developers and architects.

Document Conventions

The following notations are used to present material in this document:

ISSUE: An issue is a direct request for feedback from the audience. An issue reflects a lack of decision due to insufficient or conflicting inputs. These are resolved through the acquisition of more input

NOTE: Extra normative information that the author(s) wish to draw the attention of the reader to.

Use Cases and Requirements

The following Use cases describe the interactions supported by this specification.

1.a. Browser-based Single Sign-on to ecosystem

1. A user logs on to Jamcracker.
2. The user accesses a partner site that participates in the ITML session management ecosystem. The user is not prompted for an authentication credential. Authorization to resources is controlled by the partner. This use case may be described as centralized authentication and delegated authorization.

1.b. Log-off

1. A user has signed on to Jamcracker. They can choose to log out of the entire ecosystem from the Jamcracker portal. They can also choose to log out of a particular partner from the partner site

1.c. Time-Out

1. A user has abandoned their interactions with the ecosystem. The virtual session must be purged from all machines participating in the ecosystem and the session.

1.d. Traversal to multiple partner sites

1. A user may travel from Jamcracker to site A and Site B. It is imperative that the most recent access time of the ecosystem be the relevant access time for purposes of timeout of the ecosystem.

Requirements

1. The session information shall map easily to web sessions.
2. The algorithm must be robust in the case of failure of a communication path or node
3. The algorithm must allow for Partners to not notify in the case of session update
4. The algorithm must allow for failure of Partners
5. The algorithm must work with version 4.0 + web browsers and wireless devices
6. A user timed-out from an ASP should be able to re-access the ASP without prompt for username/password.

Assumptions

A primary assumption has been made:

1. *"Users tend to interact frequently within an ASP and infrequently interact across multiple ASPs".*

This assumption allows the use of a conversation between Jamcracker and an partner when for purposes of timing out and logging out.

2. There is a prior existing trust relationship between Jamcracker and the partner.
3. The Content of the session is defined separately, such as S2ML.
4. Offline support is not required.
5. It is required that Jamcracker be informed of accesses to partner systems for the purposes of billing, anonymity is a bug not a feature.
6. Authorization information exchange is not required.
7. Access to an ASP before access to Jamcracker is not required. That is, there is no requirement for complex redirecting from ASP to JC back to ASP if correctly logged on. Microsoft Passport, Netegrity Siteminder, etc. provide this functionality and we are loath to re-create it.

Terminology

ASP – An application provided over the net and typically charged by the month. Note that there is no technical difference between an ASP and a web site. An ASP can provide a single or multiple business processes.

ASP Integration – A platform for Collaborative Business Processes, typically offering business processes that span ASPs.

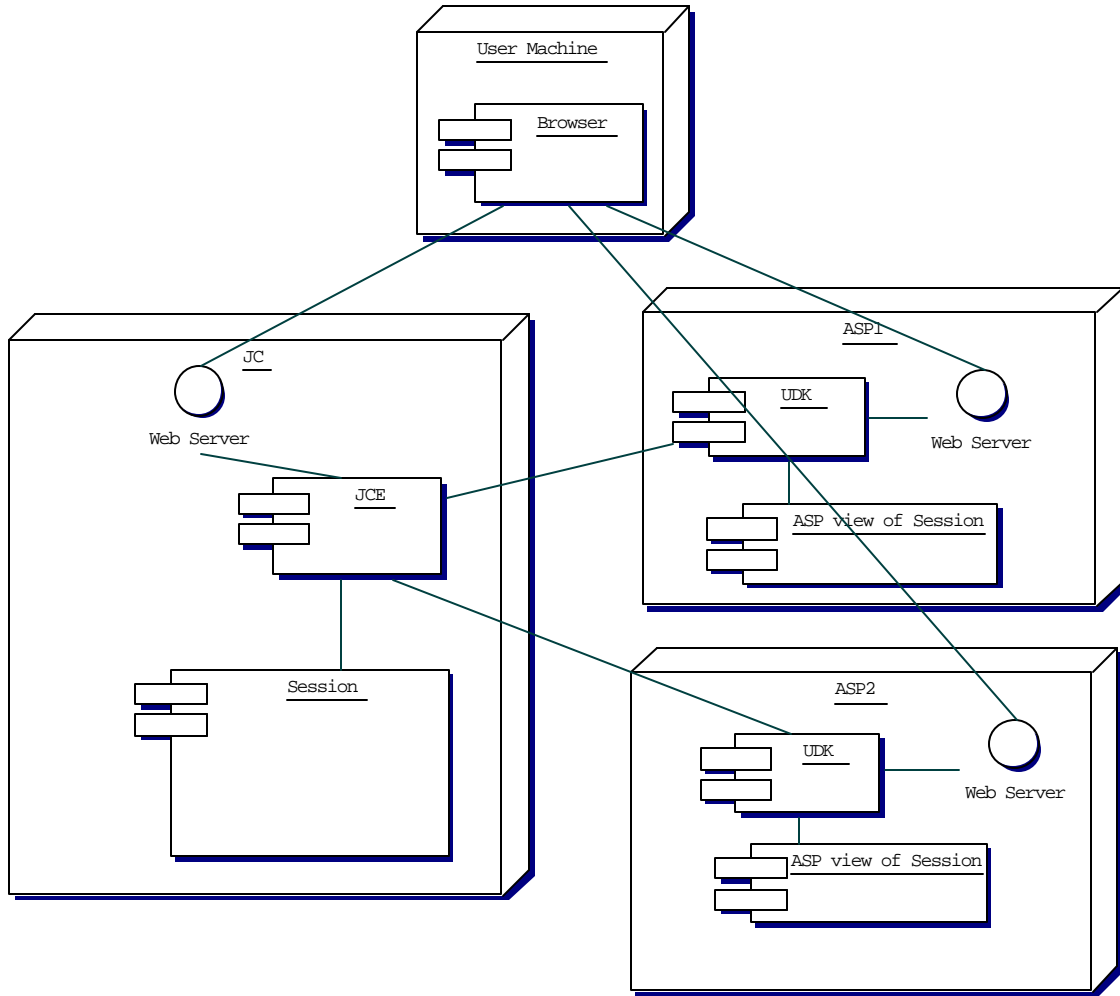
JCE – Jamcracker Enterprise. The Jamcracker server platform.

UDK – Utility Development Kit. Software that Jamcracker provides ASP partners for ASP integration

Session Management architecture

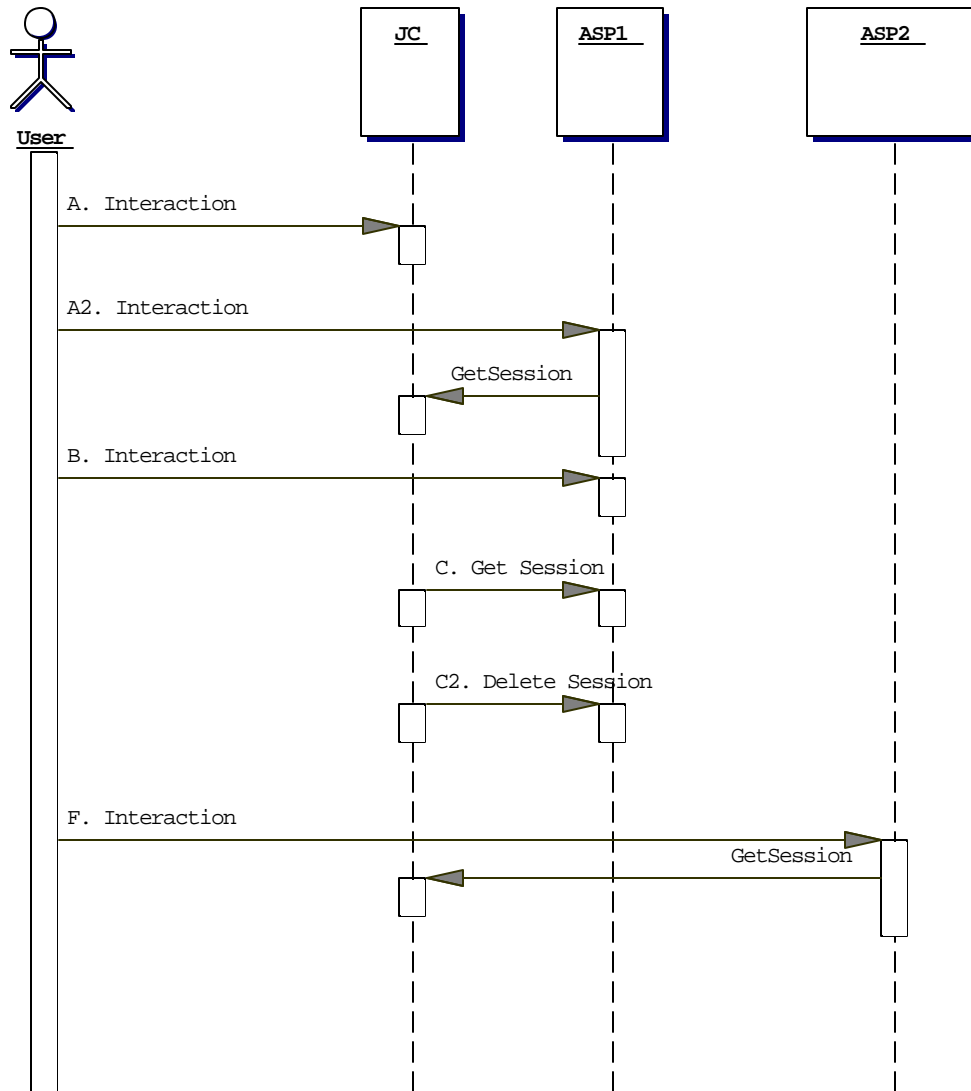
Deployment diagram

The following diagram shows the deployment of the session management:



Session Management interaction diagrams

The session management interactions are shown below



The use of Single sign-on being transferred by session management should help.

A. Simple logon and hand-off

1. A user logs onto Jamcracker. This results in the creation of a session on JC
2. The user then accesses ASP1. This results in ASP1 creating a session and then requesting the session from Jamcracker via a “getSession” message.
3. Jamcracker adds the ASP to the list of active ASPs for the given user
4. Jamcracker sends an “addSession” response
5. The user is allowed access to ASP1

B. ASP Interaction

1. User accesses ASP1 again. There is no interaction between ASP and Jamcracker.

C. JC Timeout or Logout

1. Assume that the user has gone beyond the timeout limit on JC. JC will check the active ASP session that it has attached to this user’s JC session. JC will send a “getSession” message to every ASP to determine if the user has an active session at this time.

2. If no active sessions or user has selected Logout, JC will delete the session and notify every ASP active for the user session via “deleteSession” message.

D. ASP Timeout

1. Assume that the user has gone beyond the timeout limit on ASP. The ASP logs the user out without checking with the rest of the ecosystem. Should the user request access to the ASP again, the ASP can simply follow step A2 which does not prompt the user for any information. There is no communication from ASP to Jamcracker.

E. ASP Logout

1. User requests logout from ASP. There is no communication from ASP to Jamcracker.

F. ASP1 to ASP2 hand-off

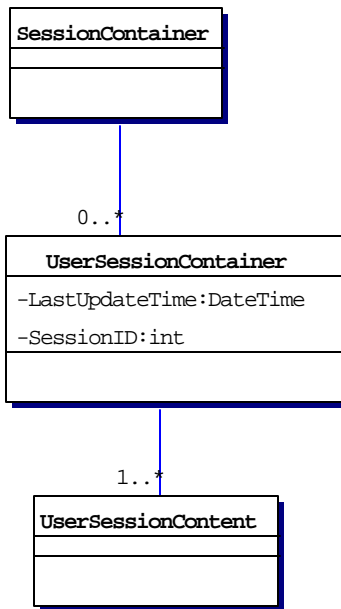
1. Starting at step A2, the user then accesses ASP2
2. ASP2 queries Jamcracker for the session.
3. JC returns the session to ASP2. There is no update to JC

Notes:

From the users perspective, the logout and timeout happen as if there was no session management. When they press logout, the ASP controls the screen. The performance impact of refreshing master and slave sessions is not seen by any end-user.

ITML Session Management Class Diagram

The class diagram for ITML Session Management follows:



The session container has a collection of user Session Containers. Each Container can have many sub-elements. A prototypical usage is where a UserSessionContent is an S2ML document.

Design guidelines

The assumption of lengthy interactions with a single ASP at a time allows the use of a Jamcracker centric polling model. Jamcracker pulls state changes from ASPs for the purposes of checking timeouts. ASPs push state changes when a logout occurs.

In total, there are 6 possible options for synchronization.

1. Jamcracker pull state whenever it needs (typically because of user request on JC or ASP). This is the design selected for timeout
2. Jamcracker pulls state at a periodic interval
3. ASP pushes state to Jamcracker whenever a change happens. This is selected for log-out.
4. ASP pushes state at a periodic interval.
5. ASP pulls state from Jamcracker whenever a potential change could occur (timeout)
6. ASP pulls state at a periodic interval

The key advantage of the Jamcracker pull approaches is that Jamcracker is not dependent upon the ASP for state management. If the ASP is down then Jamcracker will deal with the ASP being down, as opposed to not receiving an update. The key advantage for the “whenever” needed approach is that the session will be more timely – that is Jamcracker will pull the data when it needs so it will be current.

The downside of the Jamcracker pull approach is that there may be a significant performance hit when Jamcracker pulls the data from the ASP. This approach may also result in significantly increased traffic between JC and the ASP. There may be an interaction between JC and the ASP for every user on the JC ecosystem.

ASPs may push state to Jamcracker only when the session is being deleted from their system. Other updates to the state on their system are propagated only when JC pulls the data.

The time associated with each change to a session is a delta time, with time zero being the time of the last update from Jamcracker. This prevents the problem of systems having different values for the current time.

The changes propagated by the session management are the current value. The alternative is deltas or log of updates to the store. Thus add/delete/modify on various elements within the session are transmitted.

The entire session for a given user is always transmitted. It is anticipated that the data set will be fairly small (<5 Kilobytes) so optimization of transmitting subgraphs is not necessary

SessionIDs are assigned by Jamcracker. Jamcracker assures that the sessionIDs are unique across the space of ASPs it deals with.

Issues:

- Should Jamcracker combine multiple requests into a single request? For example, every request within a 1 second period is combined so there is only 1 message/second flowing.
- Is there an algorithm for update conflicts required?
- If there is an update conflict algorithm to be used on the asp, should the result be sent back to Jamcracker? Ie, update(sessionID, session) returns a new session?
- Should session store a state (like deleted) instead of deleting the session? Or is requiring that ASPs update JC on state changes (add/delete) sufficient?

Session Management Messages

Namespace

All Provisioning elements are associated with the sess namespace identifier, which is bound to the namespace URI <http://www.itml.org/ns/2001/01/sessmgmt>.

Requests

The session management specification consists of a number of commands and responses.

Jamcracker can issue the following requests to an ASP with parameters are provided:

- GetSession(SessionID) – retrieve session
- DeleteSession(SessionID) – Render user session inactive.

An ASP can issue the following requests to Jamcracker

- GetSession(UserID) – retrieve a session
- DeleteSession(SessionID).

NOTE: There is no accepted standard for defining interfaces to XML/Web services, though WSDL appears to be gaining consensus. It is expected that this specification will evolve to any standard that emerges.

Errors

The following error conditions are defined in the sessmgmt error namespace:

- InvalidSessionID
- InvalidSessionInfo
- InvalidCompanyID
- InvalidUserID

Samples

The following show fragments of sample requests and response. These are not integrated with SOAP due to current authoring tool limitations on mixing namespaces.

Sample Request - getSession

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by David Orchard (Jamcracker) -->
<!-- Instance of a fully formed getSession -->
<sess:getSession xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xmlns="http://www.itml.org/ns/2001/01/sessmgmt" xsi:noNamespaceSchemaLocation="ITMLSessMethods.xsd"
txid="abc:88:88:88:88">
  <sess:UserIdentity>
    <sess:UserID>dorchard</sess:UserID>
    <sess:CompanyID>Partner1</sess:CompanyID>
  </sess:UserIdentity>
</sess:getSession>
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by David Orchard (Jamcracker) -->
<!-- Instance of a fully formed getSessionResponse -->
<sess:getSessionResponse xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xmlns="http://www.itml.org/ns/2001/01/sessmgmt"
xsi:noNamespaceSchemaLocation="ITMLSessMethods.xsd"
txid="abc:88:88:88:88">
  <sess:UserSessionContainer>
    <lastUpdateTime>+05</lastUpdateTime>
    <SessionIdentity>alargehardtoguessstring</SessionIdentity>

    <s2ml:NameAssertion xmlns="http://www.s2ml.org">
      <This>urn:authEngine32:xsd12</This>
      <Issuer>http://www.jamcracker.com</Issuer>
      <Date>2000-10-16T12:34:120-05:00</Date>
      <Audiences>urn:jceecosystem</Audiences>
      <AuthData>
        <AuthType>Login</AuthType>
        <IdentityToken>x12+21defqa$3#</IdentityToken>
      </AuthData>
      <dsig:signature>...</dsig:signature>
    </s2ml:NameAssertion>

    <bpi:bpdata xmlns="http://www.itml.org/ns/2001/05/bpi">
      <bpi:bpdata>
        <sess:UserSessionContainer>
      </sess:UserSessionContainer>
    </bpi:bpdata>
  </sess:getSessionResponse>
```

Sample Response, error

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by David Orchard (Jamcracker) -->
<!-- Instance of a fully formed getSessionResponse -->
<sess:getSessionResponse xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xmlns="http://www.itml.org/ns/2001/01/sessmgmt" xsi:noNamespaceSchemaLocation="ITMLSessMethods.xsd"
txid="abc:88:88:88:88">
  <sess:ITMLFaultDetail>
    <sess:faultcode>InvalidUserID</sess:faultcode>
    <sess:faultstring>C'est une nomme invalide</sess:faultstring>
  </sess:ITMLFaultDetail>
</sess:getSessionResponse>
```

Appendix

Session Message Schemas

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified">
  <xsd:include schemaLocation="ITMLSessStructs.xsd"/>
  <xsd:element name="getSession">
    <xsd:complexType>
      <xsd:choice>
        <xsd:element name="UserIdentity" type="UserIdentityType"/>
        <xsd:element name="SessionIdentity" type="SessionIdentityType"/>
      </xsd:choice>
      <xsd:attribute name="txid" type="txidType"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="getSessionResponse">
    <xsd:complexType>
      <xsd:choice>
        <xsd:sequence>
          <xsd:element name="UserSessionContainer" type="UserSessionType"/>
        </xsd:sequence>
        <xsd:element name="ITMLFaultDetail" type="ITMLFaultDetailType" minOccurs="0"/>
      </xsd:choice>
      <xsd:attribute name="txid" type="txidType"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="deleteSession">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="UserIdentity" type="UserIdentityType"/>
      </xsd:sequence>
      <xsd:attribute name="txid" type="txidType"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="deleteSessionResponse">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="ITMLFaultDetail" type="ITMLFaultDetailType" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="txid" type="txidType"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Session Structure Schemas

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Edited by David Orchard, Jamcracker -->
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified">
  <xsd:simpleType name="UserIDType">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
  <xsd:simpleType name="CompanyIDType">
```

```

    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
  <xsd:simpleType name="SessionIdentityType">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
  <xsd:complexType name="UserIdentityType">
    <xsd:sequence>
      <xsd:element name="UserID" type="UserIDType"/>
      <xsd:element name="CompanyID" type="CompanyIDType"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="UserSessionContainer">
    <xsd:sequence>
      <xsd:element name="LastUpdateTime" type="xsd:timeDuration"/>
      <xsd:element name="SessionID" type="SessionIdentityType"/>
      <xsd:element name="UserSession" type="UserSessionType" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="UserSessionType">
    <xsd:sequence>
      <xsd:any namespace="##any" processContents="lax"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="ITMLFaultDetailType">
    <xsd:sequence>
      <xsd:element name="faultcode" type="faultcodeType"/>
      <xsd:element name="faultstring" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:simpleType name="faultcodeType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="InvalidUserID"/>
      <xsd:enumeration value="InvalidSessionID"/>
      <xsd:enumeration value="InvalidCompanyID"/>
      <xsd:enumeration value="InvalidSessionInfo"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="txidType">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[a-z]{3}:[0-9]{2}:[0-9]{2}:[0-9]{2}:[0-9]{2}"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>

```

Futures

-

Important Design notes (non-normative)

Error Handling and Recovery

The specification has avoided any mention of error recovery or increased reliability. Typical examples of these are retrying the transaction, adopting a two-phase commit protocol, defining compensating transactions. Therefore it is likely that this will be a trouble spot for integration.

Credits

This specification has been greatly helped by the following people and affiliations:

Revision History

2000:

Dec 8 Initial revision

Dec 12: Version 0.2, Added schema, added design notes, added messages

2001:

Jan 1: Version 0.3. Conch model – there can be only one! Active ASP at a time

Jan 15: Version 0.4. Conch model thrown out, cleaned up for publication.

Jan 30: Version 0.5. Cleaned up use cases, updated to latest XML Schema and wildcard