

labinfotech summit

INTEROP-2009

Rationale: Despite three decades of firmly entrenched information technology presence within the clinical laboratory domain, there had been:

- No emergence of a standardized, seamless & interoperable solution by which records can be exchanged between institutions, without the need for customized interfaces and programming
- No public attempts to demonstrate such interoperability at domain-appropriate events (as already has been demonstrated with Radiology and EHR systems)

“Out of the box” LIS Interoperability remains an unmet need

labinfotech summit

INTEROP-2009

- What Interop is:
 - A partnership of vendors, academia and government
 - A real-world example of cutting-edge federated web architecture at work
 - A real-world example of cloud services attached to the federation, adding incremental value to retrieved data, in the form of interpretative services
 - An interactive experience, whereby attendees of LITS can observe seamless interoperability in an actual federation of LIS, govt. and interpretive services vendors

Historical Setting:

- HL7 is not a tightly constrained standard and as such, it is not a true standard at all
- Every interface constructed with our current antiquated methods should be considered as a unique instantiation, thus requiring customized support and specialized expertise
- As the number of requisite or desirable interconnections increase within and between our enterprises, the overall healthcare system evolves towards a point of technical un-sustainability, in terms of stewardship of such systems and interfaces.
- In support of intra- and inter-institutional interfaces, what is required is the adoption and use of true standards, based upon modern data representation and exchange methods.

What are these modern methods?

- eXtensible Markup Language (XML)
- Federated architectures
- Properly adjudicated namespaces and strongly typed concepts and data elements (ISO-11179)
- Service-oriented Architectures (SOAs) and normalized data models
- Grid Computing
- Cloud Service Architectures

Our present manner of construction of conventional LIS interfaces differs from the above list in essentially every aspect.

Interop 2009 Project Design

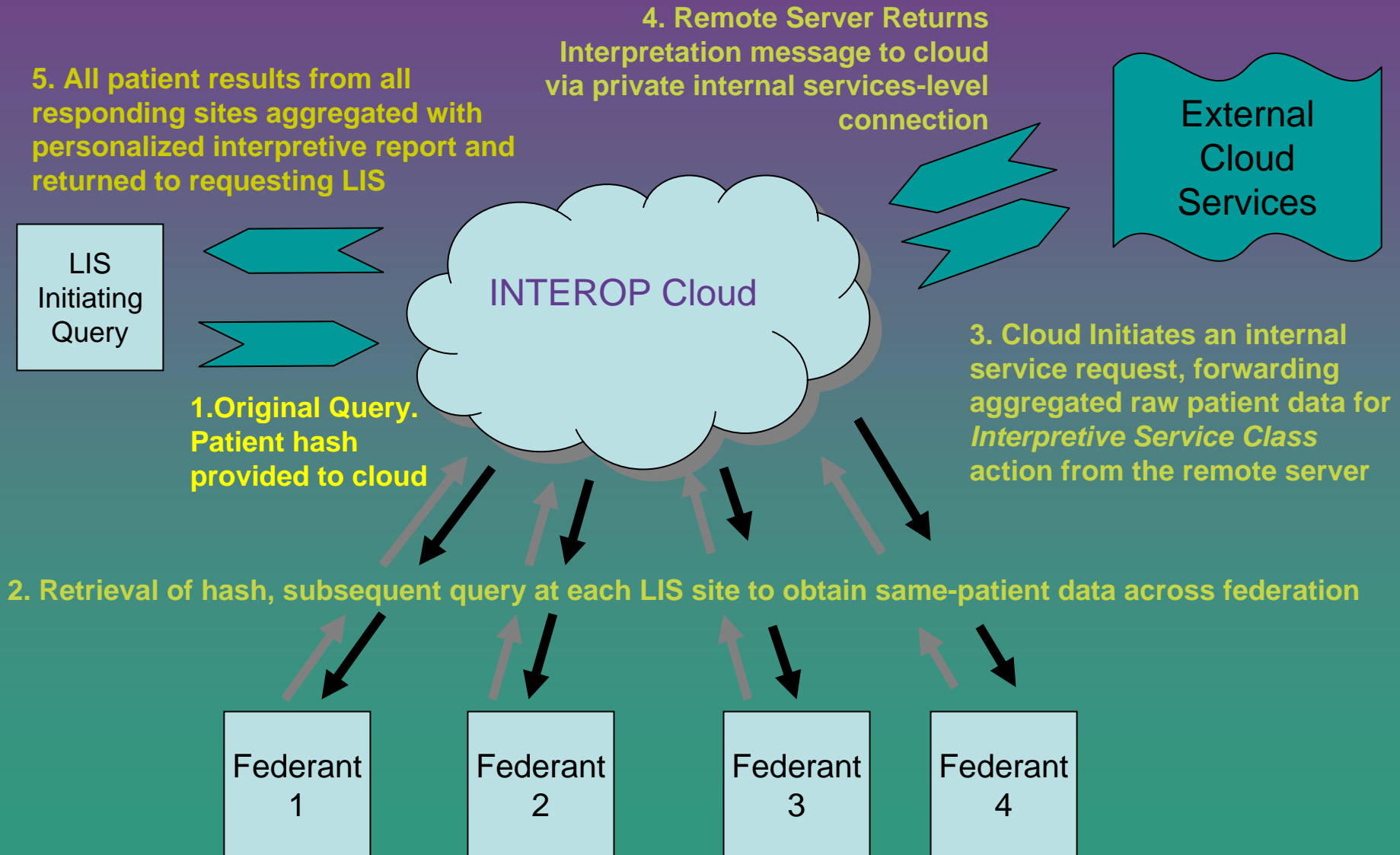
- Partnership of Industry, government and academia
 - Initially announced in March of 2008
 - At present:
 - Five major LIS vendors
 - One data interpretive services vendor
 - National Cancer Institute caBIG biospecimen repository working group
 - Dr. Ian Fore
 - University of Michigan
 - Create a grid-services solution for demonstration at LITS-2009 that would demonstrate the following:
 - Seamless interoperability of exchange of lab results between commercial LISs
 - Interactive format for LITs attendees, similar to DICOM / IHE “connecathons” of the prior two decades
 - Effort for implementation not to exceed 40 man-hours per site
 - All tools based on open-source architectures and software
 - All tools and standards to be released to the public domain
 - Core functionality to serve as the edifice for subsequent expansion of the grid services
 - Fully functional in time for LITS-2009

[Edit this page](#)[Old revisions](#)[Recent changes](#) [Search](#)

Trace: » [interop.p_hash](#) » [patient_hash_algorithm_definition](#) » [soap.interop.register](#) » [interop_api_soap](#) » [interop.get](#) » [node.get](#) » [node.accept](#) » [interop_api_xml-rpc](#) »
Start » [interop_participants](#)

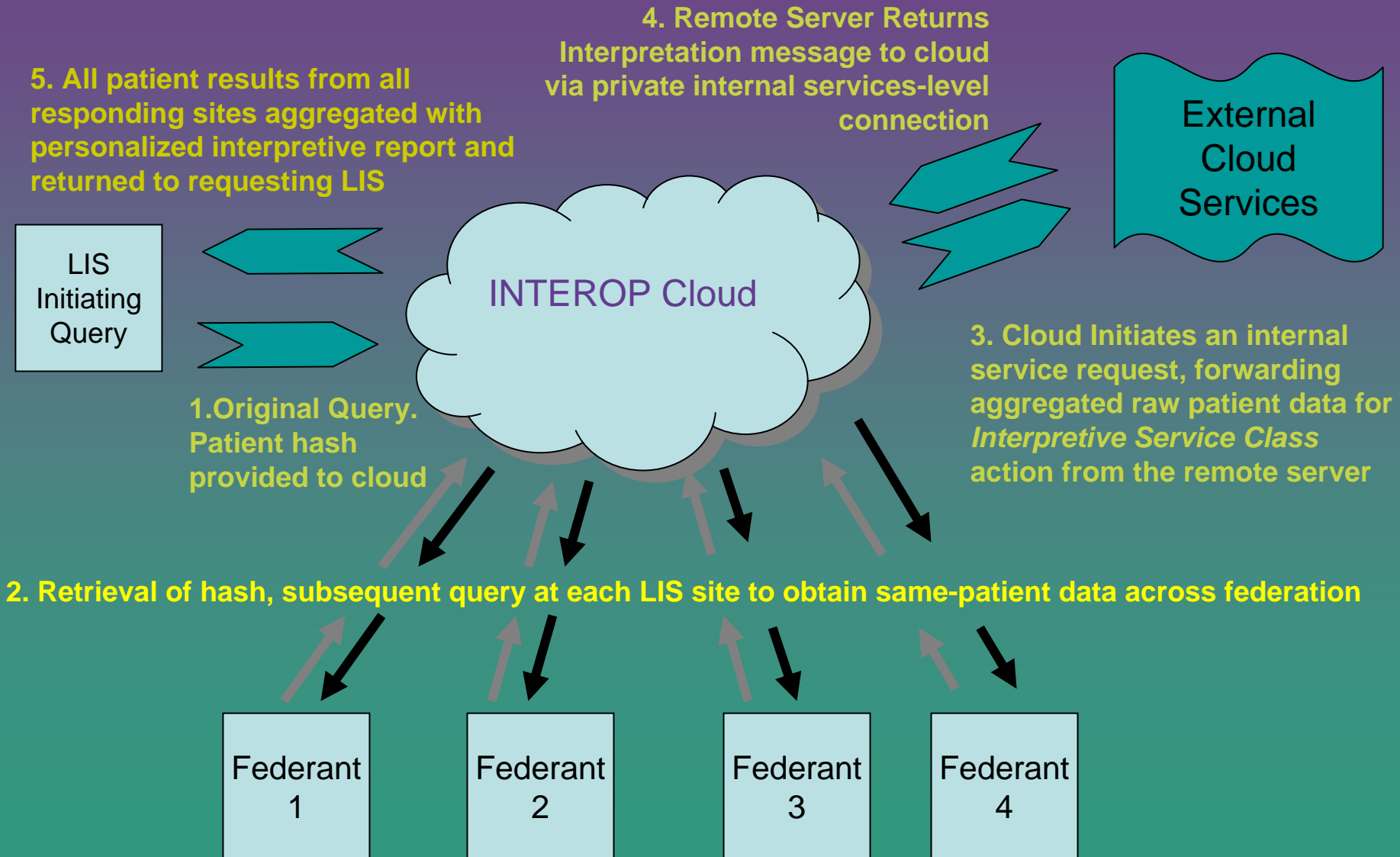
Interop Participants

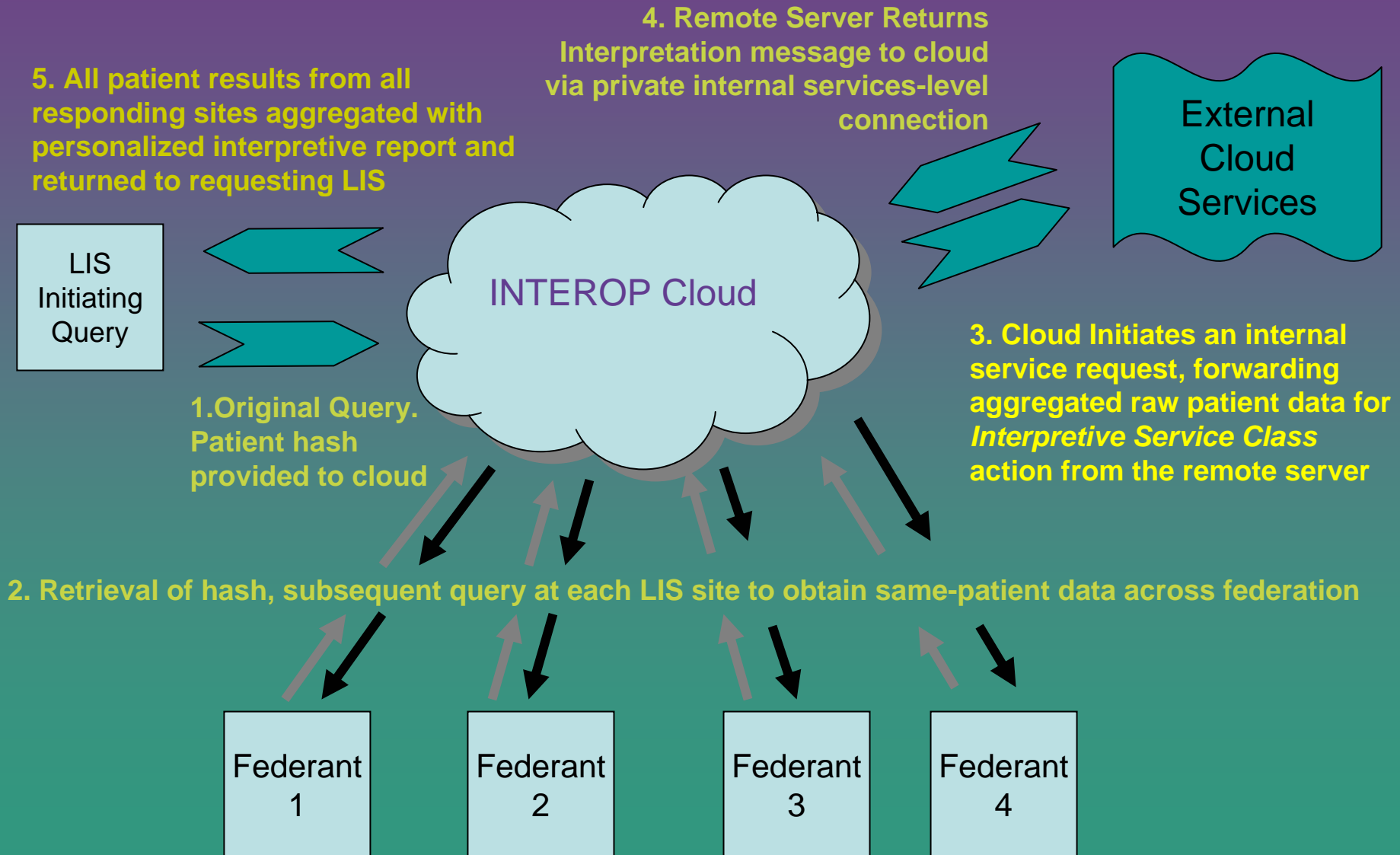
Company	Name	E-Mail
Atlas Development Corp	Jeffrey Nonhoff-Zieg	✉ jnonhoffzieg@atlasdev.com
Atlas Development Cord	Hari Viswanathan	✉ hviswanathan@atlasdev.com
McKesson	Arlene Tani	✉ Arlene.Tani@McKesson.com
McKesson	Dan Worley	✉ Dan.Worley@McKesson.com
McKesson	John Yount	✉ John.Yount@McKesson.com
McKesson	Peter Callies	✉ Peter.Callies@McKesson.com
mTuitive	John Murphy	✉ john.murphy@mtuitive.com
mTuitive	Mark B. Law	✉ mark.law@mtuitive.com
mTuitive	Peter O'Toole	✉ peter.otoole@mtuitive.com
NIH	Ian Fore	✉ forei@mail.nih.gov
NIH/Booz Allen Hamilton	Juergen Klenk	✉ klenk_juergen@bah.com
NIH/Booz Allen Hamilton	Vikram Purohit	✉ purohit_vikram@bah.com
NIH/Booz Allen Hamilton	Anna Fernandez	✉ fernandez_anna@bah.com
NIH/Emory University	Tony Pan	✉ tpan@bmi.osu.edu
Pacific Knowledge Systems	Matthew Pittman	✉ m.pittman@pks.com.au
Pacific Knowledge Systems	Rob Manser	✉ r.manser@pks.com.au
SCC	Greg Kumik	✉ gregk@softcomputer.com
SCC	Leszek Rumak	✉ rum@softcomputer.com
Sunquest Information Systems	Chris Hosier	✉ Chris.Hosier@ahsl.com
Sunquest Information Systems	David Wilson	✉ David.Wilson@ahsl.com
Sunquest Information Systems	Richard Batch	✉ Richard.batch@sunquestinfo.com
TechniData	Jacques Baudin	✉ jacques.baudin@technidata-web.com
TechniData	Laurent Lardin	✉ laurent.lardin@technidata-web.com

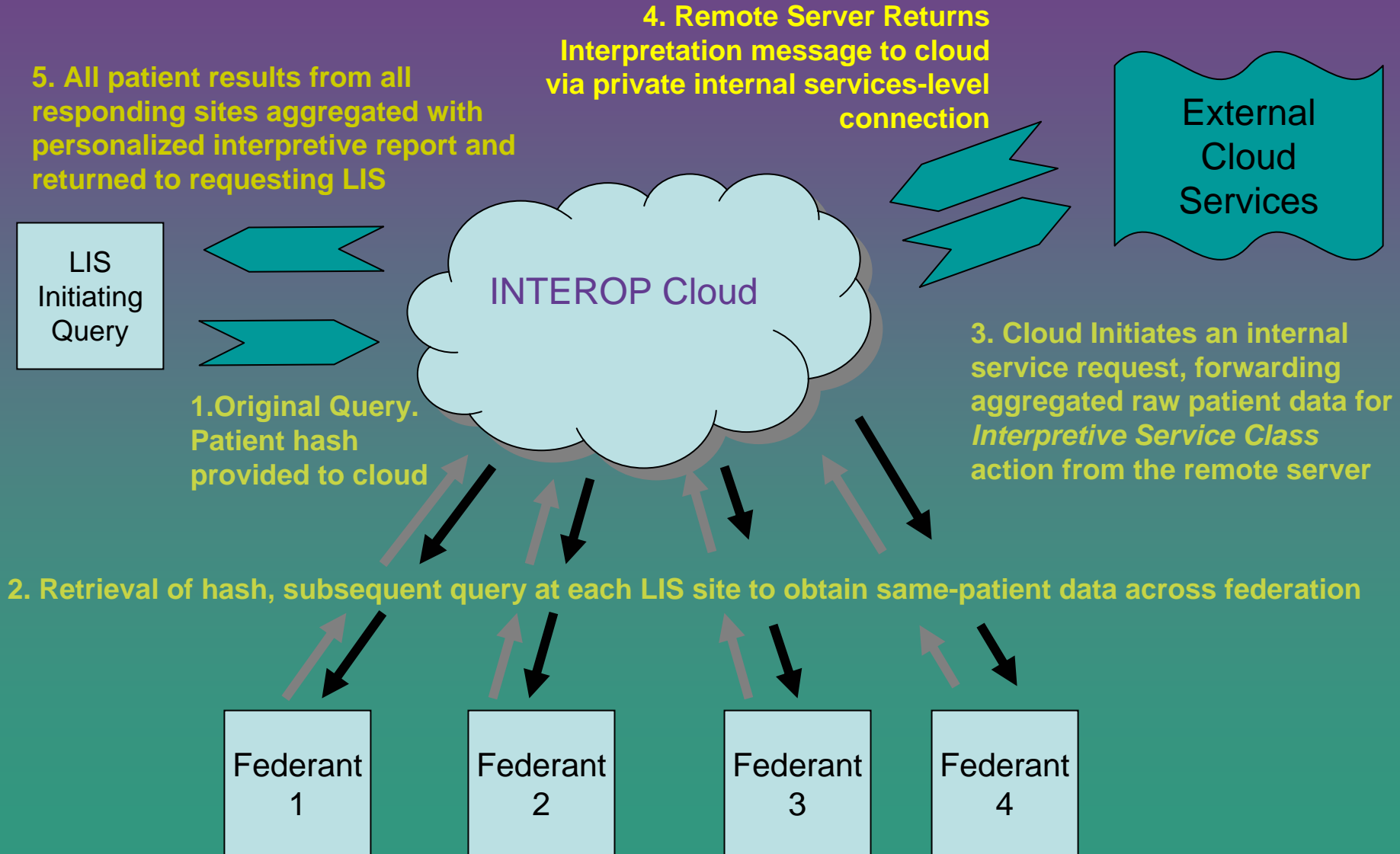


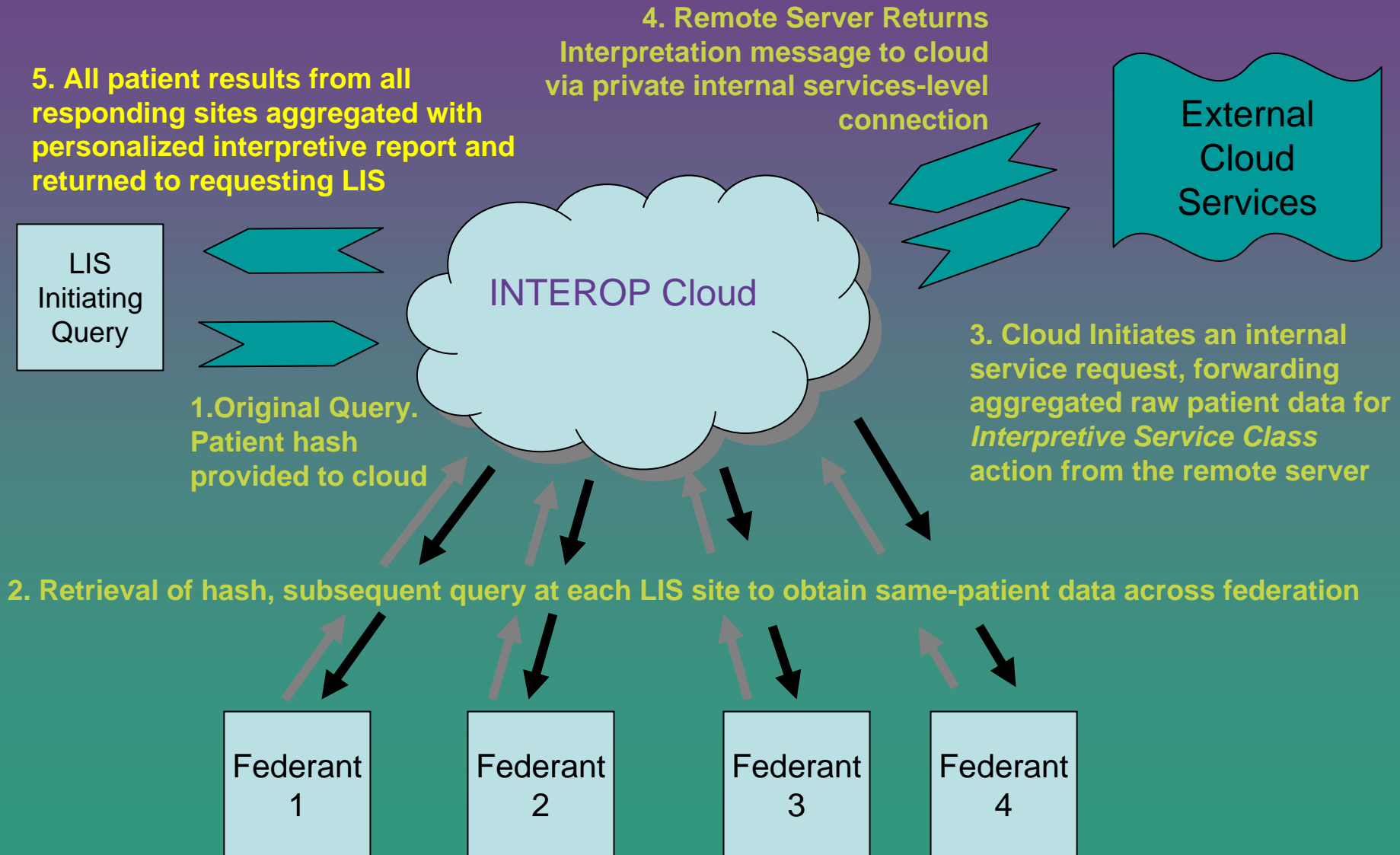
labinfotech summit

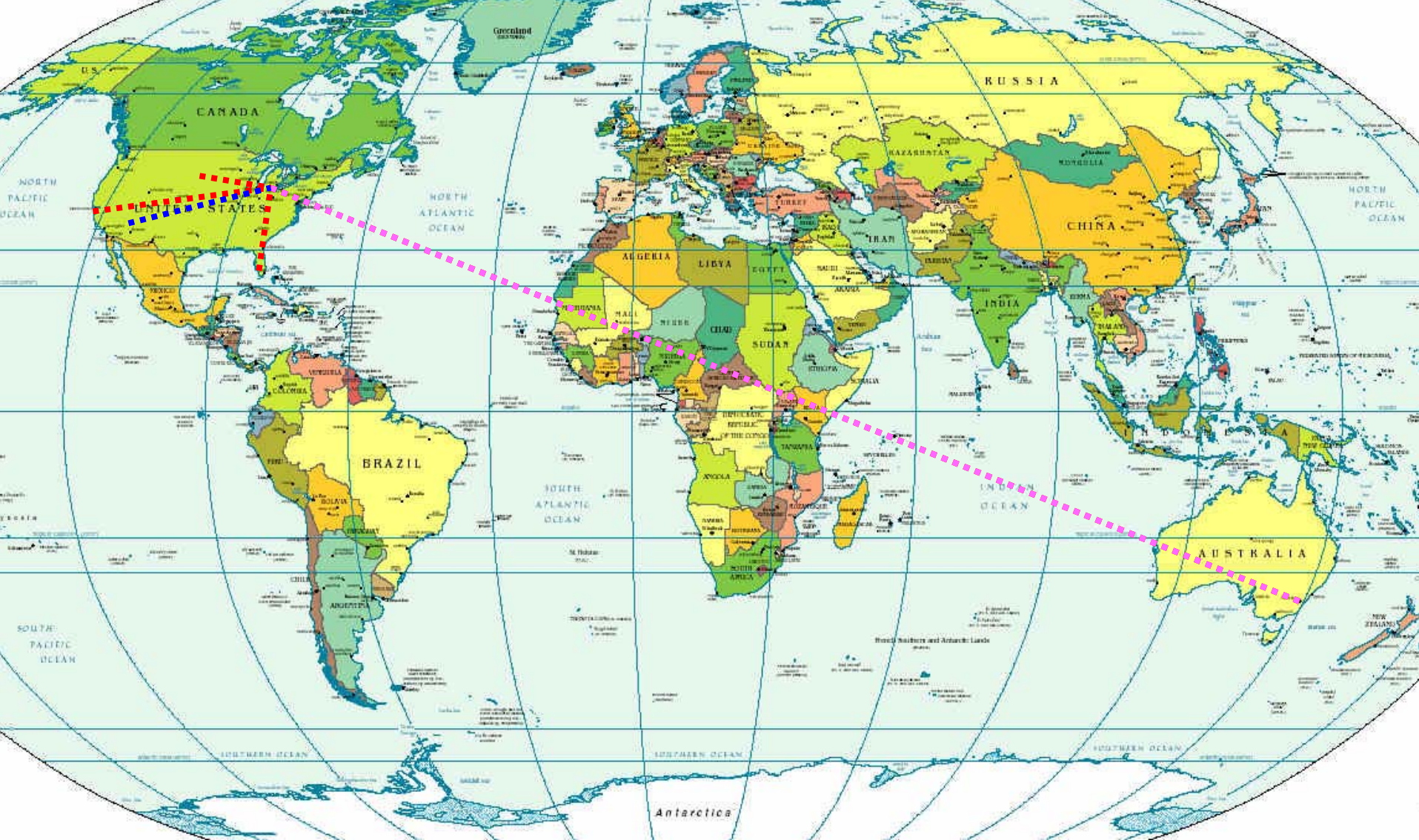
INTEROP-2009











Core Interop Cloud Server Prior to Attachment of Clients.

InterOp Server Dashboard		Synchronizing Log Data		Connected to InterOp Database 10/04/34	
LITS-Interop Server	0	xmlrpc soap	141.214.4.75	server	On-Line (log)
LITS-Node	112358	xmlrpc	https://141.214.4.75:444	(unknown)	Off-Line (log)
work_jeff	6864	xmlrpc	http://10.21.208.140:8887	(unknown)	Off-Line (log)
SCC_DEMOLAPTOP	9862002	xmlrpc	http://208.86.238.126:8080/SCCInteropNode/xmlrpcServlet	(unknown)	Off-Line (log)
Atlasdev	9862003	xmlrpc	https://interops-testing.elaborders.com/NodeListener.rem	(unknown)	Off-Line (log)
Mckesson	9862004	xmlrpc	https://173.8.114.33:8443/wo/xmlrpc	(unknown)	Off-Line (log)
SCC_SITE44	9862044	xmlrpc	http://208.86.238.90:8080/SCCInteropNode/xmlrpcServlet	(unknown)	Off-Line (log)
IMFTESTNODE	9911	SOAP	76.21.187.33	(unknown)	Off-Line (log)
9862	TESTNODE	soap	http://141.214.4.75/soap/	(unknown)	Off-Line (log)

Core Interop Cloud Server With Attached Clients.

InterOp Server Dashboard Synchronizing Log Data					
LITS-Interop Server	0	xmlrpc soap	141.214.4.75	server	Connected to InterOp Database 23:14:44 On-Line (log)
2009-03-13 22:17:22 Failed Interop ping by 76.21.187.33. Invalid session.					
LITS-Node	112358	xmlrpc	https://141.214.4.75:444	6a5f9cf720ac6d39de4fdeb6d46844a1	On-Line (log)
2009-03-13 18:58:15 Pinged Interop Successfully. 141.214.4.75					
work_jeff	6864	xmlrpc	http://10.21.208.140:8887	e3ebf0c014976d11218dc4c47e545545	On-Line (log)
2009-03-13 17:38:30 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(10.21.208.140) transaction(ea9c039d2062294bd96c49a0fbf9d585)					
SCC_DEMOLAPTOP	9862002	xmlrpc	http://208.86.238.126:8080/SCCInteropNode/xmlrpcServlet	b74503bc66643a71902e6b7e18b3b170	On-Line (log)
2009-03-13 17:38:43 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(ea9c039d2062294bd96c49a0fbf9d585) to 208.86.238.126					
Atlasdev	9862003	xmlrpc	https://interop-testing.elaborders.com/NodeListener.rem	3ec092226dc29cb7ba83d7d991fcacce	On-Line (log)
2009-03-13 17:38:34 Checksum mismatch with the data from site(207.178.204.152) on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(ea9c039d2062294bd96c49a0fbf9d585)					
Mckesson	9862004	xmlrpc	https://173.8.114.33:8443/wo/xmlrpc	8f078d18e1403d67584bc9aa629b0ba0	On-Line (log)
2009-03-13 17:38:36 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(ea9c039d2062294bd96c49a0fbf9d585)					
SCC_SITE44	9862044	xmlrpc	http://208.86.238.90:8080/SCCInteropNode/xmlrpcServlet	c4ff42762e5f46daef275ab643f78c27	On-Line (log)
2009-03-13 17:39:58 208.86.238.90 successfully pinged 9862044					
9911	IMFTESTNODE	soap	http://76.21.187.33/soap/	(unknown)	Off-Line (log)
2009-03-09 19:00:06 Node Registered - Site(9911), Key(IMFTESTNODE), IP(76.21.187.33), url(http://76.21.187.33/soap/), protocol(soap)					

InterOp Server Dashboard Synchronizing Log Data

LITS-Interop Server

0

xmlrpc|soap

141.214.4.75

server

Connected to InterOp Database 23:17:16

On-Line (log)

```
2009-03-09 15:39:09 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:39:07 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:39:06 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:39:05 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:39:05 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:39:04 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:39:03 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:39:03 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:39:02 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:39:02 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:39:01 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:39:01 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:39:00 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:39:00 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:38:59 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:38:59 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:38:58 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:38:58 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
2009-03-09 15:38:58 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f8bed86f4c9bad21a2176be180c397eb) to 81.86.230.68
```

InterOp Server Dashboard Synchronizing Log Data

LITS-Interop Server

0

xmlrpc|soap

141.214.4.75

server

Connected to InterOp Database 23:18:26

On-Line (log)

```
2009-03-06 14:57:17 Data hashed for patient 1504c2063d03520b7ac03310217c8979
2009-03-06 14:57:12 Data hashed for patient 3a972b71974d4671465f4574838cd676
2009-03-06 14:56:57 Data hashed for patient baea807c11f7df1543f848c82bdb0ca9
2009-03-06 14:56:48 Data hashed for patient b18ad95ac5c11f07598a984ec8682beb
2009-03-06 14:56:42 Data hashed for patient 52bc8da14f5364cfc2738d87abdd09ee
2009-03-06 14:56:37 Data hashed for patient bc53849f53648c5857314291fd91037f
2009-03-06 14:56:32 Data hashed for patient 2f88312d9ac32a2738b76f26670cf063
2009-03-06 14:56:27 Data hashed for patient 0a6fb937a25b80959149906a5bcaa453
2009-03-06 14:56:22 Data hashed for patient 36ddced1d1b277db61cf242bdb57c7de
2009-03-06 14:56:22 Failed Login. 208.86.238.90 has an existing session.
2009-03-06 14:56:17 Data hashed for patient 76bab36cbf8add89e22d7072da0397ac
2009-03-06 14:56:17 Node Registered - Site(SCC_SITE), Key(9862002), IP(208.86.238.90), url(https://208.86.238.90/), protocol(xmlrpc)
2009-03-06 14:56:06 Data hashed for patient 2e6cbd17982d5ce9a45c6aa92f3686ed
2009-03-06 14:56:06 Node Unregistered - Key: 9862002
2009-03-06 14:56:02 Data hashed for patient 49a5c542a0f6e356645213692b3d0ddb
2009-03-06 14:56:01 Node Unregistered - Key: 9862002
2009-03-06 14:55:57 Data hashed for patient bf9fa897488de63c30483da07821ffaa
2009-03-06 14:55:52 Data hashed for patient e8275de0edf46ed83ce98e21b9dcb48c
2009-03-06 14:55:51 Failed Login. 208.86.238.90 has an existing session.
```

InterOp Server Dashboard Synchronizing Log Data

LITS-Interop Server

0

xmlrpc|soap

141.214.4.75

server

Connected to InterOp Database 23:19:40

On-Line (log)

```
2009-03-06 07:38:30 Data hashed for patient dabd67560c7030a7dd6ad831af403689
2009-03-06 07:38:30 Data hashed for patient 60d6bf232bc1c4133ca5aa56c21599cb
2009-03-06 07:38:29 Data hashed for patient 8c4087778f8b51fac85faf4a2d8d4463
2009-03-06 07:38:29 Data hashed for patient 76225cc4df4f51ac41f84bebd2f62006
2009-03-06 07:38:29 Data hashed for patient 39d57e64a07ba20825b2280ab5f0cecc
2009-03-06 07:38:28 Data hashed for patient bd18c5ad307a877beb33e10035da35fb
2009-03-06 07:38:28 Data hashed for patient cfbadc4c8e095d40d090046f88923bff
2009-03-06 07:38:27 Data hashed for patient a9887875f8b545aae386efc951881a54
2009-03-06 07:38:27 Data hashed for patient 6f237bed33f0d740320b3e713ce96071
2009-03-06 07:38:26 Data hashed for patient 0f8d6c82026fb6200ac8a3d50cc0a032
2009-03-06 07:38:26 Data hashed for patient 096e658c5c620b85dd0bce9ae6e48ecd
2009-03-06 07:37:39 Sending data on patient(b98fcb66c4976cfe43ab5f1c905f7efa) transaction(37f346e4a0a4df790b1d083713df7ce3) to 208.86.238.90
2009-03-06 07:37:38 The node 143.112.32.4 is unreachable and being logged out
2009-03-06 07:37:36 Sending data on patient(b98fcb66c4976cfe43ab5f1c905f7efa) transaction(5a364e413198d10f9bbf9ba77a6624a) to 208.86.238.90
2009-03-06 07:37:35 The node 143.112.32.4 is unreachable and being logged out
2009-03-06 07:37:34 Failed Interop ping by 79.123.32.146. Invalid session.
2009-03-06 07:37:26 Data hashed for patient 63aa4c6fc5c9fbd2aa9a2093e005058f
2009-03-06 07:37:25 Data hashed for patient f965b232fc3267d906bb3a9b1a66e787
2009-03-06 07:37:25 Data hashed for patient 90486248057474d3dfba384b29bb9c24
```

InterOp Server Dashboard Synchronizing Log Data

LITS-Interop Server

0

xmlrpc|soap

141.214.4.75

server

Connected to InterOp Database 23:20:37

On-Line (log)

```
2009-03-05 10:44:47 Node Registered - Site(SCC_SITE), Key(9862002), IP(208.86.238.90), url(https://208.86.238.90/), protocol(xmlrpc)
2009-03-05 10:41:08 Failed Interop ping by 208.86.238.90. Invalid session.
2009-03-05 10:40:58 Node Unregistered - Key: 9862002
2009-03-05 10:40:49 Failed Login. 208.86.238.90 has an existing session.
2009-03-05 10:32:40 Node Registered - Site(SCC_SITE), Key(9862002), IP(208.86.238.90), url(https://208.86.238.90/), protocol(xmlrpc)
2009-03-05 10:07:49 Node Unregistered - Key: 9862002
2009-03-05 10:07:48 Successful Logout for session 9862002
2009-03-05 10:07:48 Data requested for patient b98fcb66c4976cfe43ab5f1c905f7efa by 208.86.238.90. Transaction ID: 5d64a87569eeb047ad00a9436898305a
2009-03-05 10:07:48 208.86.238.90 could not ping 9862002: Invalid Site Key
2009-03-05 10:07:47 Failed Interop ping by 208.86.238.90. Invalid session.
2009-03-05 09:53:24 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(892da786ed83a4c9cee90576dff34f8c) to 141.214.4.75
2009-03-05 09:53:24 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(892da786ed83a4c9cee90576dff34f8c) to 141.214.4.75
2009-03-05 09:49:47 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(6b3c841c7e046d1a4847e61f175c1df9) to 141.214.4.75
2009-03-05 09:49:27 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(0eb6ce92b3999c3be071f750d868d0df) to 141.214.4.75
2009-03-05 09:48:25 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(5d77a6ae7252a013378f11041b3e02b8) to 141.214.4.75
2009-03-05 09:45:47 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(246393eebe82101deb75dbbda1f67148) to 141.214.4.75
2009-03-05 09:42:11 The node 141.214.4.75 is unreachable and being logged out
2009-03-05 09:42:11 Requesting node 141.214.4.75 is unreachable. Transaction(acc3936e3e88ad897920b8eb9311c4eb)
2009-03-05 09:42:11 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(acc3936e3e88ad897920b8eb9311c4eb) to 141.214.4.75
```

InterOp Server Dashboard		Synchronizing Log Data						Connected to InterOp Database 10:02:10	
LITS-Interop Server	0	xmlrpc soap	141.214.4.75	server				On-Line	(log)
2009-03-05 11:08:11 Failed Interop ping by 141.214.4.75. Invalid session.									
LITS-Node	112358	xmlrpc	https://141.214.4.75:444		6a5f9cf720ac6d39de4fdeb6d46844a1			On-Line	(log)
2009-03-14 07:58:54 Pinged Interop Successfully.141.214.4.75									
work_jeff	6864	xmlrpc	http://10.21.208.140:8887		e3ebf0c014976d11218dc4c47e545545			On-Line	(log)
2009-03-14 07:50:15 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(10.21.208.140) transaction(6da799e2d6b88e97b57169133aa8790c)									
SCC_DEMOLAPTOP	9862002	xmlrpc	http://208.86.238.126:8080/SCCInteropNode/xmlrpcServlet		b74503bc66643a71902e6b7e18b3b170			On-Line	(log)
<p>2009-03-13 17:38:43 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(ea9c039d2062294bd96c49a0fb9d585) to 208.86.238.126</p> <p>2009-03-13 17:38:42 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(ea9c039d2062294bd96c49a0fb9d585) to 208.86.238.126</p> <p>2009-03-13 17:36:17 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(096d7a5ce5150d682240f682df3389f9) to 208.86.238.126</p> <p>2009-03-13 17:36:17 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(096d7a5ce5150d682240f682df3389f9) to 208.86.238.126</p> <p>2009-03-13 17:21:54 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(208.86.238.126) transaction(ee013a5fed0d64e78dfe8576edbeeb58a)</p> <p>2009-03-13 17:21:21 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(208.86.238.126) transaction(0cb02c9e6e5d3c204893142fa01b4980)</p> <p>2009-03-13 17:18:03 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(208.86.238.126) transaction(530bb7aa2873d888f476abee58cafd7d)</p> <p>2009-03-13 17:17:51 Fetching data on patient(cc63f3f6515a087d3da3155aaacc326a) from site(208.86.238.126) transaction(b702dbae91b41141ff549909b8751378)</p> <p>2009-03-13 17:17:15 208.86.238.126 successfully pinged 9862002</p> <p>2009-03-13 17:17:13 Successful Login from 208.86.238.126</p> <p>2009-03-13 17:17:11 Node Registered - Site(SCC_DEMOLAPTOP), Key(9862002), IP(208.86.238.126), url(http://208.86.238.126:8080/SCCInteropNode/xmlrpcServlet), protocol(xmlrpc)</p> <p>2009-03-13 17:17:08 Node Unregistered - Key: 9862002</p> <p>2009-03-13 17:16:06 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(859f8ba523cfd92ccaef39f073766b7) to 208.86.238.126</p> <p>2009-03-13 17:16:06 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(859f8ba523cfd92ccaef39f073766b7) to 208.86.238.126</p> <p>2009-03-13 17:14:23 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(3a47c7b849d7737767d2b3e3cbee9a4a) to 208.86.238.126</p> <p>2009-03-13 17:14:23 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(3a47c7b849d7737767d2b3e3cbee9a4a) to 208.86.238.126</p> <p>2009-03-13 17:06:40 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f3dd6070d16c7453f539249cffcd53d3) to 208.86.238.126</p> <p>2009-03-13 17:06:39 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(f3dd6070d16c7453f539249cffcd53d3) to 208.86.238.126</p> <p>2009-03-13 14:48:28 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(4cb922b4e4c3cad1c96b50c3b1b80846) to 208.86.238.126</p>									
Atlasdev	9862003	xmlrpc	https://interop-testing.elaborders.com/NodeListener.rem		3ec092226dc29cb7ba83d7d991fcacce			On-Line	(log)
2009-03-13 11:12:16 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(207.178.204.152) transaction(58fd89656242bf737ae49665ce1effcb)									
Mckesson	9862004	xmlrpc	https://173.8.114.33:8443/wo/xmlrpc		8f078d18e1403d67584bc9aa629b0ba0			On-Line	(log)
2009-03-13 14:23:54 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(b939d3419e812d34265eb272daaba89c)									
SCC_SITE44	9862044	xmlrpc	http://208.86.238.90:8080/SCCInteropNode/xmlrpcServlet		c4ff42762e5f46daef275ab643f78c27			On-Line	(log)
2009-03-14 07:50:31 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(208.86.238.90) transaction(6da799e2d6b88e97b57169133aa8790c)									
9911	IMFTESTNODE	soap	http://76.21.187.33/soap/		(unknown)			Off-Line	(log)
2009-03-14 07:42:40 Node Registration Updated - Site(76.21.187.33), Key(IMFTESTNODE), IP(76.21.187.33), url(http://76.21.187.33), protocol(soap)									

InterOp Server Dashboard

Synchronizing Log Data

Connected to InterOp Database 23:21:42

LITS-Interop Server

0

xmlrpc|soap

141.214.4.75

server

On-Line (log)

2009-03-05 10:44:47 Node Registered - Site(SCC_SITE), Key(9862002), IP(208.86.238.90), url(https://208.86.238.90/), protocol(xmlrpc)

LITS-Node

112358

xmlrpc

https://141.214.4.75:444

6a5f9cf720ac8d39de4fdeb6d46844a1

On-Line (log)

2009-03-13 18:58:15 Pinged Interop Successfully.141.214.4.75

work_jeff

6864

xmlrpc

http://10.21.208.140:8887

e3ebf0c014976d11218dc4c47e545545

On-Line (log)

2009-03-13 17:38:30 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(10.21.208.140) transaction(ea9c039d2062294bd96c49a0fb9d585)

SCC_DEMOLAPTOP

9862002

xmlrpc

http://208.86.238.126:8080/SCCInteropNode/xmlrpcServlet

b74503bc66643a71902e6b7e18b3b170

On-Line (log)

2009-03-13 17:38:43 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(ea9c039d2062294bd96c49a0fb9d585) to 208.86.238.126

Atlasdev

9862003

xmlrpc

https://interop-testing.elaborders.com/NodeListener.rem

3ec092226dc29cb7ba83d7d991fcacce

On-Line (log)

2009-03-13 17:38:34 Checksum mismatch with the data from site(207.178.204.152) on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(ea9c039d2062294bd96c49a0fb9d585)

Mckesson

9862004

xmlrpc

https://173.8.114.33:8443/wo/xmlrpc

8f078d18e1403d67584bc9aa629b0ba0

On-Line (log)

2009-03-13 17:38:36 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(ea9c039d2062294bd96c49a0fb9d585)

2009-03-13 17:36:10 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(096d7a5ce5150d682240f882df3389f9)

2009-03-13 17:21:47 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(ee013a5fed0d64e78dfe8576edbeeb58a)

2009-03-13 17:21:14 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(0cb02c9e6e5d3c204893142fa01b4980)

2009-03-13 17:17:56 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(530bb7aa2873d888f476abee58cafd7d)

2009-03-13 17:17:46 Fetching data on patient(cc63f3f6515a087d3da3155aaacc326a) from site(173.8.114.33) transaction(b702dbae91b41141ff549909b8751378)

2009-03-13 17:15:56 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(859f8ba523cfd92ccae139f073766b7)

2009-03-13 17:15:31 Fetching data on patient(437e9560692b6ae5639143864b7c6656) from site(173.8.114.33) transaction(524dd0a8915e77c1ce3e2f020feb365)

2009-03-13 17:14:13 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(3a47c7b849d7737767d2b3e3cbee9a4a)

2009-03-13 17:09:16 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(ec629050e115ebfe6dd88c79d99244ae)

2009-03-13 17:08:32 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(74894aee72c9f72cb0beb6169103397)

2009-03-13 17:06:59 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(124e41c87cd670643898b3599807569c)

2009-03-13 17:06:30 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(f3dd6070d16c7453f539249cffcd53d3)

2009-03-13 17:02:53 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(bf46de66a8459aa4c52f00b8a1f67f86)

2009-03-13 17:02:00 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(5a915d128b3f2d4d158c18774c2ed5f9)

2009-03-13 16:57:24 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(63c4fe6298250625f80f403822063204)

2009-03-13 16:43:01 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(e9bf2d9285bf10612e46f2b0f4949243) to 173.8.114.33

2009-03-13 16:43:00 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(e9bf2d9285bf10612e46f2b0f4949243) to 173.8.114.33

2009-03-13 16:42:58 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(66d8eb50710d5c122b2adde9eac47460) to 173.8.114.33

SCC_SITE44

9862044

xmlrpc

http://208.86.238.90:8080/SCCInteropNode/xmlrpcServlet

c4ff42762e5f46daef275ab643f78c27

On-Line (log)

2009-03-13 17:39:58 208.86.238.90 successfully pinged 9862044

9911

IMFTESTNODE

soap

http://76.21.187.33/soap/

(unknown)

Off-Line (log)

2009-03-09 19:00:06 Node Registered - Site(9911), Key(IMFTESTNODE), IP(76.21.187.33), url(http://76.21.187.33/soap/), protocol(soap)

InterOp Server Dashboard

Synchronizing Log Data

Connected to InterOp Database 23:25:19

LITS-Interop Server

0

xmlrpc|soap

141.214.4.75

server

On-Line (log)

2009-03-05 10:44:47 Node Registered - Site(SCC_SITE), Key(9862002), IP(208.86.238.90), url(https://208.86.238.90/), protocol(xmlrpc)

LITS-Node

112358

xmlrpc

https://141.214.4.75:444

6a5f9cf720ac6d39de4fdeb6d46844a1

On-Line (log)

2009-03-13 18:58:15 Pinged Interop Successfully.141.214.4.75

work_jeff

6864

xmlrpc

http://10.21.208.140:8887

e3ebf0c014976d11218dc4c47e545545

On-Line (log)

2009-03-13 17:38:30 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(10.21.208.140) transaction(ea9c039d2062294bd96c49a0fb9d585)

SCC_DEMOLAPTOP

9862002

xmlrpc

http://208.86.238.126:8080/SCCInteropNode/xmlrpcServlet

b74503bc66643a71902e6b7e18b3b170

On-Line (log)

2009-03-13 17:38:43 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(ea9c039d2062294bd96c49a0fb9d585) to 208.86.238.126

Atlasdev

9862003

xmlrpc

https://interop-testing.elaborders.com/NodeListener.rem

3ec092226dc29cb7ba83d7d991fcacce

On-Line (log)

2009-03-13 17:38:34 Checksum mismatch with the data from site(207.178.204.152) on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(ea9c039d2062294bd96c49a0fb9d585)

Mckesson

9862004

xmlrpc

https://173.8.114.33:8443/wo/xmlrpc

8f078d18e1403d67584bc9aa629b0ba0

On-Line (log)

2009-03-13 14:12:10 Data validation complete on patient(70eadb4c1869f3cdc8e4c649f258c9a) transaction(4eeb4aab3deb3d1dd0f4a1646b5e86b7) to 173.8.114.33

2009-03-13 14:12:10 Sending data on patient(70eadb4c1869f3cdc8e4c649f258c9a) transaction(4eeb4aab3deb3d1dd0f4a1646b5e86b7) to 173.8.114.33

2009-03-13 14:06:56 Data validation complete on patient(138061946808cbb90110a9f9ff200afe) transaction(8cfe01bddd875027fc65748b2958fbab) to 173.8.114.33

2009-03-13 14:06:56 Sending data on patient(138061946808cbb90110a9f9ff200afe) transaction(8cfe01bddd875027fc65748b2958fbab) to 173.8.114.33

2009-03-13 14:03:36 Data validation complete on patient(89550b9019d6d46bca7a98c2819b21e8) transaction(a07bca997c8d0209400251d9ac6b5801) to 173.8.114.33

2009-03-13 14:03:36 Sending data on patient(89550b9019d6d46bca7a98c2819b21e8) transaction(a07bca997c8d0209400251d9ac6b5801) to 173.8.114.33

2009-03-13 14:01:10 Data validation complete on patient(89550b9019d6d46bca7a98c2819b21e8) transaction(e742bf1a93dcee3d88c851383defa0d9) to 173.8.114.33

2009-03-13 14:01:09 Sending data on patient(89550b9019d6d46bca7a98c2819b21e8) transaction(e742bf1a93dcee3d88c851383defa0d9) to 173.8.114.33

2009-03-13 13:55:23 Data validation complete on patient(89550b9019d6d46bca7a98c2819b21e8) transaction(39466bf67589e4cd38171ef79df31484) to 173.8.114.33

2009-03-13 13:55:22 Sending data on patient(89550b9019d6d46bca7a98c2819b21e8) transaction(39466bf67589e4cd38171ef79df31484) to 173.8.114.33

2009-03-13 13:40:03 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(38e384547a24cee1a57c33a58018ad2f)

2009-03-13 13:11:55 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(173.8.114.33) transaction(a05facb06acbec9d1126b829846db7cd)

2009-03-13 13:07:28 Fetching data on patient(750e339e2c70ba262ba19aa121d75bd3) from site(173.8.114.33) transaction(2cc06a3d3e8490aea3161e5f58ccaa14)

2009-03-13 13:03:18 Fetching data on patient(750e339e2c70ba262ba19aa121d75bd3) from site(173.8.114.33) transaction(38d432e37dd38050c08dd08200b9a771)

2009-03-13 13:03:03 Fetching data on patient(2c46cb4a0a0f813ee66280d5cacc7b5e) from site(173.8.114.33) transaction(6e5f0a145b3e3d79b6ed285cfff48d9)

2009-03-13 13:02:51 Fetching data on patient(2c46cb4a0a0f813ee66280d5cacc7b5e) from site(173.8.114.33) transaction(69f075aaf0f4008213c926d72d4b84f)

2009-03-13 13:02:24 Fetching data on patient(5b8d759b540177acbd7a6be277f5d135) from site(173.8.114.33) transaction(5db87e168f128fae28340ff9a9bae1bb)

2009-03-13 13:01:59 Fetching data on patient(eced81c4842f52f7a11308280c74aca) from site(173.8.114.33) transaction(b1dc2b8af4c7ff1314d874ddee960a94)

2009-03-13 13:01:35 Fetching data on patient(5b8d759b540177acbd7a6be277f5d135) from site(173.8.114.33) transaction(a4208b3ad09878f6407c48263621bde9)

SCC_SITE44

9862044

xmlrpc

http://208.86.238.90:8080/SCCInteropNode/xmlrpcServlet

c4ff42762e5f46daef275ab643f78c27

On-Line (log)

2009-03-13 17:39:58 208.86.238.90 successfully pinged 9862044

9911

IMFTESTNODE

soap

http://76.21.187.33/soap/

(unknown)

Off-Line (log)

2009-03-09 19:00:06 Node Registered - Site(9911), Key(IMFTESTNODE), IP(76.21.187.33), url(http://76.21.187.33/soap/), protocol(soap)

InterOp Server Dashboard

Synchronizing Log Data

Connected to InterOp Database 23:27:56

LITS-Interop Server

0

xmlrpc|soap

141.214.4.75

server

On-Line (log)

2009-03-05 10:44:47 Node Registered - Site(SCC_SITE), Key(9862002), IP(208.86.238.90), uri(https://208.86.238.90/), protocol(xmlrpc)

LITS-Node

112358

xmlrpc

https://141.214.4.75:444

6a5f9cf720ac6d39de4fdeb6d46844a1

On-Line (log)

2009-03-13 18:58:15 Pinged Interop Successfully.141.214.4.75

work_jeff

6864

xmlrpc

http://10.21.208.140:8887

e3ebf0c014976d11218dc4c47e545545

On-Line (log)

2009-03-13 17:38:30 Fetching data on patient(d0124d1a11a1ca44877a60ba900a50b7) from site(10.21.208.140) transaction(ea9c039d2062294bd96c49a0fb9d585)

SCC_DEMOLAPTOP

9862002

xmlrpc

http://208.86.238.126:8080/SCCInteropNode/xmlrpcServlet

b74503bc66643a71902e6b7e18b3b170

On-Line (log)

2009-03-13 17:38:43 Data validation complete on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(ea9c039d2062294bd96c49a0fb9d585) to 208.86.238.126

Atlasdev

9862003

xmlrpc

https://interop-testing.elaborders.com/NodeListener.rem

3ec092226dc29cb7ba83d7d991fcacce

On-Line (log)

2009-03-12 18:38:21 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(03b0c7722bf3ee8af72b8ab8de0bd991) to 207.178.204.152

2009-03-12 18:26:14 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(7aab1167ee93c9fb8af46d5a7bf9822c) to 207.178.204.152

2009-03-12 18:19:48 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(e27e10ea8886a8ab0ed1e478496a2393) to 207.178.204.152

2009-03-12 18:12:31 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(5e9db12fa4cffca7a79ce35e215a2d72) to 207.178.204.152

2009-03-12 18:12:24 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(797f9bd8f5d2c23632c3285d183a765f) to 207.178.204.152

2009-03-12 18:12:23 Pinged Interop Successfully.207.178.204.152

2009-03-12 18:12:14 Successful Login from 207.178.204.152

2009-03-12 15:55:17 Sending data on patient(af8e399347849d4f5534f0a1b99d481b) transaction(e9fb47ba3ad2a77fa2df5a67970455e5) to 207.178.204.152

2009-03-12 15:06:37 Sending data on patient(af8e399347849d4f5534f0a1b99d481b) transaction(3d633787f1c7d6ae510fbc8835782667) to 207.178.204.152

2009-03-12 15:04:49 Sending data on patient(d0124d1a11a1ca44877a60ba900a50b7) transaction(68b6ba491e2e4b35297e020551f0925) to 207.178.204.152

2009-03-12 15:04:16 Node Registered - Site(Atlasdev), Key(9862003), IP(207.178.204.152), uri(https://interop-testing.elaborders.com/NodeListener.rem), protocol(xmlrpc)

2009-03-12 15:04:11 Node Unregistered - Key: 9862003

2009-03-12 15:03:38 Node Registered - Site(Atlasdev), Key(9862003), IP(207.178.204.152), uri(https://interop-testing.elaborders.com/NodeListener.rem), protocol(xmlrpc)

2009-03-12 15:03:33 Node Unregistered - Key: 9862003

2009-03-12 14:52:14 Successful Login from 207.178.204.152

2009-03-12 14:52:08 Node Registered - Site(Atlasdev), Key(9862003), IP(207.178.204.152), uri(https://interop-testing.elaborders.com/NodeListener.rem), protocol(443)

2009-03-12 14:48:03 Node Unregistered - Key: 9862003

2009-03-12 14:46:33 Failed Login. 207.178.204.152 has an existing session.

2009-03-12 14:46:22 Node Registered - Site(Atlasdev), Key(9862003), IP(207.178.204.152), uri(http://interop-testing.elaborders.com/NodeListener.rem), protocol(443)

Mckesson

9862004

xmlrpc

https://173.8.114.33:8443/wo/xmlrpc

8f078d18e1403d67584bc9aa629b0ba0

On-Line (log)

2009-03-13 14:12:10 Data validation complete on patient(70eadb4c1869f3cdcf8e4c649f258c9a) transaction(4eeb4aab3deb3d1dd0f4a1646b5e86b7) to 173.8.114.33

SCC_SITE44

9862044

xmlrpc

http://208.86.238.90:8080/SCCInteropNode/xmlrpcServlet

c4ff42762e5f46daef275ab643f78c27

On-Line (log)

2009-03-13 17:39:58 208.86.238.90 successfully pinged 9862044

9911

IMFTESTNODE

soap

http://76.21.187.33/soap/

(unknown)

Off-Line (log)

2009-03-09 19:00:06 Node Registered - Site(9911), Key(IMFTESTNODE), IP(76.21.187.33), uri(http://76.21.187.33/soap/), protocol(soap)

InterOp Node Dashboard

LITS-Node

141.214.4.75

112358

2009-03-14 08:53:30

Connected to InterOp Database 09:56:56

(On-Line)

(log)

Report Date/Time: 02/06/09 0918 Brooks, Dorothy L Age: 76 YRS
(0000)XXXX-XXX-X DOB: 12/18/32,)(Brooks, Dorothy L (0000)XXXX-XXX-X 12345678901234567

University of Michigan Sample 03/10/09

FEMALE Brooks, Dorothy L
76 YRS (0000)XXXX-XXX-X
12/18/32

[CHEMICAL PATHOLOGY]

COLLECTION DATE 02/09/09
TIME 0500

REFERENCE UNITS

SAMPLE LIPID PANEL

Sex F

Sample_type U

Cholesterol 171.6 0.0-200.0 mg/dl

HLD_chol 50.7 40.0-60.0 mg/dl

LDL_chol 74.1 0.0-130.0 mg/dl

Triglycerides 231.4 H 0.0-150.0 mg/dl

TC_HDL_ratio 3.4

ALP 65.0 30.0-130.0 U/L

AST 22.0 8.0-30.0 U/L

L = LOW, H = HIGH

End of Report

DOCTOR, ANY

Brooks, Dorothy L
02/09/09 03/18/09 1337 (0000)XXXX-XXX-X

InterOp Node Dashboard

LITS-Node

141.214.4.75

112358

2009-03-14 08:53:30

Connected to InterOp Database 09:50:52

(On-Line)

[log](#)

Pacific Knowledge Systems Subscription Results:

```
(clientInterpretation){
approved = False
caseId = "b04ee61e28a750a5b49932923370d68c"
kbCode = "LP"
report = "Analytes
Chol mg/dL 171.6
Trig mg/dL 231.4
LDL mg/dL 74.1
HDL mg/dL
```

Current Episode

Mild hypertriglyceridaemia due to increases in chylomicrons and/or VLDL. This pattern can be seen in non-fasting or post-prandial states.

Ensure patient has fasted 9 to 12 hours prior to performig lipid studies."

```
reportSections[] =
```

```
(reportSections){
```

```
commentAfterBlankLine = False
```

```
commentOnNewLine = True
```

```
commentsWithLinks[] =
```

```
(commentsWithLinks){
```

```
text = "Mild hypertriglyceridaemia due to increases in chylomicrons and/or VLDL. This pattern can be seen in non-fasting or post-prandial states."
```

```
},
```

```
(commentsWithLinks){
```

```
text = "Ensure patient has fasted 9 to 12 hours prior to performig lipid studies."
```

```
},
```

```
firstCommentStartedOnSameLineAsHeader = False
```

```
heading = "Current Episode"
```

```
headingAfterBlankLine = True
```

```
headingShown = True
```

```
},
```

```
site = "UMHS"
```

```
suppressed = False
```

```
}
```

```
==== End of Transaction =====
```



LITS - Interop Wiki Home

Table of Contents

- LITS - Interop Wiki Home
- Table of Contents
- Discussion





Current InterOp Version	1.5
Wiki RSS Feed	 http://141.214.4.75/feed.php
Glance Session	 http://umichpath.glance.net
	(Session Key is announced in the conference call)
InterOp Server Dashboard	 http://141.214.4.75/interop.php ← Middle of rework - doesnt work so well right now
InterOp Node Dashboard	 http://141.214.4.75/node.php

Table of Contents

1. InterOp Overview and Intended Attendee Workflow Model

Explanation of the LITS InterOp and the demonstration workflow for the participants at the conference.

I. Background

II. Activities already completed

III. Activities to be completed by Vendor Participants, prior to the meeting

IV. Activities to be completed by Lab Infotech Interop Staff, at Registration, On site

V. Activities to be carried out by Vendor Participants, in the course of demonstrating the intact functionality of the cloud

VI. Scripted Attendee Use of Cloud

a. Model one (pre-populated data)

b. Model one (pre-populated data)

VII. Note

2. InterOp Participant Contact List

Listing of Contact Information for the LITS InterOp Participants.

3. InterOp Development Team Contact List

Listing of Contact Information for the LITS InterOp Developers.

4. Development Task List & Scratch Pad

Note Pad and Task List for the Development Team. Add suggestions for next revisions in the discussion section of this page.

5. InterOp Randomized Patient Demographic File

InterOp Patient Demographic Files. Hash values are available for Q/A of the Patient Hash function.

6. InterOp API XML-RPC

API calls for XML-RPC clients. Definitions and example XML Request and Response messages

7. InterOp API SOAP

API calls for SOAP clients. Functionality is currently under development.

8. Patient Hash Algorithm Definition

Definition of the Patient Hash algorithm with example of a hashed demographic.

InterOp Overview and Intended Attendee Workflow Model

Table of Contents

- InterOp Overview and Intended Attendee Workflow Model
 - Background
 - Activities already completed
 - Activities to be completed by Vendor Participants, prior to the meeting
 - Activities to be completed by Lab Infotech Interop Staff, at Registration, On site
 - Activities to be carried out by Vendor Participants, in the course of demonstrating the intact functionality of the cloud
 - Scripted Attendee Use of Cloud
 - Model one (pre-populated data)
 - Model two (newly-populated data)
 - Note

Background

The central mission of this anticipated demonstration is the creation of an interactive experience, whereby attendees of the 2009 Lab Infotech Summit will witness seamless exchange and aggregation of laboratory results across the plurality of participating vendors (including, we hope, a functioning node sponsored by National Cancer Institute, using their Cancer Bioinformatics Grid (CABig) services architecture as the "LIS Equivalent," as NCI has expressed its intent to participate as a federant).

Upon local registration at the conference, each attendee will be given a 3×5 card which will have one of the names from the first 400 entries of the present list of 1000 randomized names. As we do not anticipate more than 400 attendees, this should be adequate for allowing a satisfactory demonstration of the cloud's correct operation, for all that are in attendance.

Because of time constraints and the potential for slight variation in textual data entry, it was felt that creating a pre-coordinated list of randomized patient demographics was preferable to expecting participants to go through the drudgery of their own time-consuming data entry with each and every vendor participant, as an enabling condition for being able to see one's own data aggregated across the cloud. However, this pre-population of demographics is in no way disingenuous towards the goal of demonstrating service oriented architecture (SOA) interoperability, as indeed, each LIS vendor will have already iterated through the process of generating a local hash number from the locally-entered patient demographics. To be clear, the hash number is in the existing spreadsheet for validation purposes only. *The intent is that each and every attached node obtains a hash number for each of their mock patients by use of the services component of the cloud and not by merely entering the hash from the spreadsheet.*

At the end of the hash population process, each vendor will have, locally, a SOA-derived hash number in their local schema that will be the same number as present in all the remaining participant LIS schemata, for each respective mock patient.

Activities already completed

- A list of 1000 randomized patient demographic entries has been created and is available on the WIKI.
- A reference hash number of each patient entry has been placed in the above list, for validation purposes only.
- A *local reference node* has been activated at the University of Michigan, with all 1000 patients being present and available for query via the hash key. Also this key has been generated for each patient entry by the `interop.p_hash` service capability of the cloud.
- The first 50 patient entries of this list have had "results data" placed into the *local reference node*, guaranteeing that any query towards one of these 50 patients, by any attached node, will result in a non-null query response, with at least the contents of our *local reference node*.

Activities to be completed by Vendor Participants, prior to the meeting

Given the availability of a working cloud and an available reference node, which is configured to return lab results for the first fifty patients, it is hoped that there is an adequate test bed in place for vendors to complete two key deliverables, well ahead of the date of the Lab InfoTech Summit Meeting:

1. Fully test their attached functionality
2. Populate at least the first 400 patient entries, in their own systems, **with at least one lab result per patient**, thus guaranteeing that at the live presentation in Las Vegas, every one of these first 400 patients will have representative data on each participant system.

Completion of the above two critical tasks will guarantee that every query will obtain federated data from every other federant on the cloud, making for a successful and satisfying experience for meeting attendees.

In summary, every vendor participant is asked to:

- Ensure that at least the first 400 of the 1000 randomized patient identities are locally entered in their local system

Development Task List / Scratch Pad

Table of Contents
• Development Task List / Scratch Pad
• InterOp Server Version 1.5
• InterOp Dashboard Version 1.5
• Authenticated Dashboard
• Public Dashboard

InterOp Server Version 1.5

- Implement Subscription Service Architecture
- Create interface to add site keys for new sites (At the moment it's just me making up a number and handing it to them - The next number will be 9862003)
- ~~Change the Interop GET function so it does not attempt to get information from the requesting node~~ - implemented in dev ver 1.5 (js)
- Follow up and troubleshoot why the interop console screen occasionally throws SSL cert errors.
- ~~Add support for different Ports~~ - implemented in dev ver 1.5 (js)
- Add management dashboard / activity viewer via a web portal
- ~~Create sample data for testing get function (js)~~
- ~~Update demographic file with Hash for Q/A (jh)~~
- ~~Create wiki listing of InterOp participants (jh)~~
- Visio diagrams of workflow, Request and Response traffic

InterOp Dashboard Version 1.5

Public vs Authenticated

Note: Check out PDO in [PHP](#)

Authenticated Dashboard

- Generate Site Keys
- View Log Files
- Trigger the Server to PING a node
- Trigger the Server to Log Out a node
- Trigger the Server to Unregister a node
- Add 'Services'
- Add 'Node Subscriptions for Services'

Public Dashboard

- View registered Nodes
- View Log In/Out Status
- View 'Recent Activity' Time Stamp

Discussion

- Place at least one "result" in their local system for each of the 400 patient entries

Upon completion of the above steps, vendors are asked to carry out a final phase of pre-meeting testing and validation of their query/response functionality, such that each node receives not only other results from the cloud, but also their own results (recognizing that the collective behavior of the cloud is one of aggregation). Hence, upon submission of a query (in the form of merely a hash number) to the cloud, the cloud should return results present for all sites actively attached.

Activities to be completed by Lab Infotech Interop Staff, at Registration, On site

- Each Attendee, at the time of local registration, will be given a 3 x 5 card with their "demographics."
- The overall intended LITS-Interop attendee browsing approach will be described at the didactic presentation, providing enough information for attendees to be aware of a) the intended steps that they will need to carry out with each of the respective vendor participants and b) the underpinnings and significance of what they will be witnessing in the course of their perusing the integrated functionality of each of the available vendor LIS demonstration locations.

Activities to be carried out by Vendor Participants, in the course of demonstrating the intact functionality of the cloud

Each vendor participant will be provided internet access by the event organizers, in the form of either WPA-based WIFI or a local RJ45 100/1000MBit connection. Because of variability with respect to computational performance and plug-in capabilities, we ask that vendors be responsible for bringing to the Summit an appropriate set of computers, that are known to be compatible with the vendor's own web-based applications. We ask that at least two workstations as a minimum be provided by each vendor, with three units being the optimum goal, to allow for simplified concurrent perusing by LITS-Interop attendees.

Scripted Attendee Use of Cloud

Model one (pre-populated data)

- The attendee will present himself/herself to the vendor with his/her "identity" on the 3x5 card.
- The vendor application specialist will demonstrate the patient lookup capabilities of the local system, confirming that the patient indeed has been pre-registered at the local system
- The vendor application specialist will demonstrate local search capabilities of the LIS, obtaining all locally-present pre-resulted lab data in that local system.
- The vendor application specialist will demonstrate federated, real-time search capabilities across the cloud, obtaining all data from all attached federants, and presenting in, seamlessly, in the local LIS as an aggregated cumulative summary of all local and external data.

Model two (newly-populated data)

- The attendee will present himself/herself to the vendor with his/her "identity" on the 3x5 card.
- The attendee will ask the vendor application specialist to order a test in the local system and then result that test with appropriate contents, followed by confirmation that the result has indeed posted to that patient's record on the local system.
- The attendee will then visit the remaining vendors and demonstrate to his/her satisfaction that the result that was previously posted on the foreign system does indeed appear on the current local system, when a federated query is run against the cloud with the presented demographics of that patient's identity.

Note

Activity carried out in support of Model Two is particularly important to the success of the demonstration, as it will provide conclusive evidence to attendees that the cloud is operating in support of true, real-time federation.

Discussion

Scripted Attendee Use of Cloud

Model One (pre-populated data)

- The attendee will present himself/herself to the vendor with his/her “identity” on the 3x5 card.
- The vendor application specialist will demonstrate the patient lookup capabilities of the local system, confirming that the patient indeed has been pre-registered at the local system
- The vendor application specialist will demonstrate local search capabilities of the LIS, obtaining all locally-present pre-resulted lab data in that local system.
- The vendor application specialist will demonstrate federated, real-time search capabilities across the cloud, obtaining all data from all attached federants, and presenting in, seamlessly, in the local LIS as an aggregated cumulative summary of all local and external data.

Model Two (newly-populated data)

- The attendee will present himself/herself to the vendor with his/her “identity” on the 3x5 card.
- The attendee will ask the vendor application specialist to order a test in the local system and then result that test with appropriate contents, followed by confirmation that the result has indeed posted to that patient’s record on the local system.
- The attendee will then visit the remaining vendors and demonstrate to his/her satisfaction that the result that was previously posted on the initial foreign system does indeed appear on the current local system, when a federated query is run against the cloud with the presented demographics of that patient’s identity.

Note

Activity carried out in support of Model Two is particularly important to the success of the demonstration, as it will provide conclusive evidence that the cloud is operating in support of true, real-time federation.

XML-RPC API Calls

This section will identify and define the XML-RPC calls needed to programmatically communicate with the LITS InterOp. This model is language independent as all communication will be XML based.

Sending and receiving of data stream can be implemented where appropriate independent of platform.

Function Definition	XML-RPC Request	XML-RPC Response
interop.register	request	response
interop.unregister	request	response
interop.login	request	response
interop.logout	request	response
interop.status	request	response
interop.nodeStatus	request	response
interop.p_hash	request	response
interop.get	request	response
node.status	request	response
node.accept	request	response
node.get	request	response

Discussion

interop.register

Table of Contents

- [interop.register](#)
- [Definition](#)
- [Request](#)
- [Response](#)

Definition

This function will allow a node to register with the InterOp server. Registration Key is provided to client pre-node registration and is entered into node for transmission to InterOp. All communications are via SSL and XML-RPC. Once registered, the server retains the node in the database of registered nodes.

Node sends the "Registration Key" along with the assigned "Site Name", "URL" which is fully qualified with port, and "Protocol" (soap|xmlrpc) to the InterOp. InterOp server will require data transmissions from the nodes to come from specific IP addresses. InterOp server will also require nodes to transmit from predetermined IP addresses. On receipt of data, the InterOp server validates the information and returns a 'TRUE' or 'FALSE' depending if data validates successfully.

Request

```
<?xml version="1.0"?>
<xmlRPC>
  <methodCall>
    <methodName>register</methodName>
    <params>
      <param>
        <value><string>[SITE_NAME]</string></value>
        <!-- Human Readable Site Name -->
      </param>
      <param>
        <value><string>[REG_KEY]</string></value>
        <!-- Provided Registration Key -->
      </param>
      <param>
        <value><string>[URL]</string></value>
        <!-- Fully qualified URL of the node including port -->
      </param>
      <param>
        <value><string>[PROTOCOL]</string></value>
        <!-- soap| xmlrpc-->
      </param>
    </params>
  </methodCall>
</xmlRPC>
```

Response

```
<?xml version="1.0"?>
<xmlRPC>
  <methodResponse>
    <params>
      <param>
        <value><boolean>[TRUE|FALSE]</boolean></value>
      </param>
    </params>
  </methodResponse>
</xmlRPC>
```

Discussion

interop.unregister

Table of Contents ▲

- interop.unregister
- Definition
- Request
- Response

Definition

This function will allow a registered node to unregister with the InterOp server. The node will be removed from the servers node listing.

Node sends the registration key to the server. On receipt and validation, the server removes the node from the server's node listing. A response of 'TRUE' is returned when the node is unregistered. If the node is unable to unregister, a response of 'FALSE' is returned.

Request

```
<?xml version="1.0"?>
<xmlRPC>
  <methodCall>
    <methodName>unregister</methodName>
    <params>
      <param>
        <value><string>[REG_KEY]</string></value>
      </param>
    </params>
  </methodCall>
</xmlRPC>
```

Response

```
<?xml version="1.0"?>
<xmlRPC>
  <methodResponse>
    <params>
      <param>
        <value><boolean>[TRUE | FALSE]</boolean></value>
      </param>
    </params>
  </methodResponse>
</xmlRPC>
```

Discussion

interop.login

Table of Contents ▲
• interop.login
• Definition
• Request
• Response

Definition

This function will allow a registered node login to the InterOp server. A logged in status equates to the node being available to return data upon a request.

Node sends its registration key to the InterOp server where it is validated. Upon successful validation, the server generates a session key and returns it to the node. At that time, the InterOp server retains it as an active node and will propagate requests to the logged in node. If validation fails, a response of 'FALSE' is returned to the node.

Request

```
<?xml version="1.0"?>
<xmlRPC>
  <methodCall>
    <methodName>login</methodName>
    <params>
      <param>
        <value><str>[REG_KEY]</str></value>
      </param>
    </params>
  </methodCall>
</xmlRPC>
```

Response

```
<?xml version="1.0"?>
  <methodResponse>
    <params>
      <param>
        <value><str>[SESSION_KEY|FALSE]</str></value>
      </param>
    </params>
  </methodResponse>
</xmlRPC>
```

Discussion

interop.logout

Table of Contents

- interop.logout
- Definition
- Request
- Response

Definition

This function will allow a registered node to logout of the application. A logged out status equates to the node being unavailable to return data upon a request.

Node sends its session key to the InterOp server where it is validated. Upon a successful validation, the server flags the node as inactive and will not propagate requests to the node and a response of 'TRUE' is returned. If validation fails, a response of 'FALSE' is returned and the node remains logged in.

Request

```
<?xml version="1.0"?>
<xmlRPC>
  <methodCall>
    <methodName>logout</methodName>
    <params>
      <param>
        <value><str>[SESSION_KEY]</str></value>
      </param>
    </params>
  </methodCall>
```

Response

```
<?xml version="1.0"?>
  <methodResponse>
    <params>
      <param>
        <value><boolean>[TRUE | FALSE]</boolean></value>
      </param>
    </params>
  </methodResponse>
</xmlRPC>
```

Discussion

interop.status

Table of Contents

- interop.status
- Definition
- Request
- Response

Definition

This function allows the node to send a 'ping' to the server. The server will return a status response along with a timestamp. If the 'OK' response is returned, the node remains active for queries. Otherwise, the node is logged out.

The node sends a PING request to the InterOp server. The server sends its status along with a timestamp to the node. 'ERROR' response can be returned if the node is having problems communicating. A null response assumes there are connectivity issues.

Request

```
<?xml version="1.0"?>
<xmlRPC>
  <methodCall>
    <methodName>status</methodName>
    <params>
      <param>
        <value><string>PING</string></value>
      </param>
    </params>
  </methodCall>
</xmlRPC>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value><str>[OK|DOWN|ERROR]</str></value>
    </param>
    <param>
      <value><dateTime.iso8601>YYYYMMDDTHHMMSS</dateTime.iso8601></value>
    </param>
  </params>
</methodResponse>
</xmlRPC>
```

Discussion

interop.nodeStatus

Table of Contents

- interop.nodeStatus
- Request
- Response

This function allows a registered node to send a 'ping' to another node to verify and update the target node status on the InterOp server.

A node transmits a PING request along with the "Site Key" of the target node to the InterOp server. The InterOp passes the request to the target node. The target node responds, updating the InterOp server. The InterOp server then sends the status of the target node to the requesting node.

Request

```
<?xml version="1.0"?>
<xmlRPC>
  <methodCall>
    <methodName>nodeStatus</methodName>
    <params>
      <param>
        <value><str>PING</str></value>
        <value><str>[SITE_KEY]</str></value>
      </param>
    </params>
  </methodCall>
</xmlRPC>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value><str>[SITE_KEY]</str></value>
    </param>
    <param>
      <value><str>TRUE|FALSE</str></value>
    </param>
    <param>
      <!-- If OK -->
      <value><dateTime.iso8601>YYYYMMDDTHHMMSS</dateTime.iso8601></value>
      <!-- If ERROR -->
      <value><str>[REASON FOR ERROR]</str></value>
    </param>
  </params>
</methodResponse>
</xmlRPC>
```

Discussion

interop.p_hash

Table of Contents

- interop.p_hash
- Definition
- Request
- Response

Definition

This function allows the server to transmit patient demographics to the server and a have the hash returned.

Node transmits demographics to server where it is hashed using a salt. The patient hash is returned to the node and the hash stored as a key to identify the patient on future query requests.

Request

```
<?xml version="1.0"?>
<xmlRPC>
  <methodCall>
    <methodName>p_hash</methodName>
    <params>
      <param>
        <value><str>[FIRST_NAME]</str></value> <!-- <= 25 char -->
      </param>
      <param>
        <value><str>[LAST_NAME]</str></value> <!-- <= 32 char -->
      </param>
      <param>
        <value><str>[MIDDLE_INITIAL]</str></value> <!-- <= 1 char -->
      </param>
      <param>
        <value><str>[SUFFIX]</str></value> <!-- <= 10 char -->
      </param>
      <param>
        <value><str>[PREFIX]</str></value> <!-- <= 10 char -->
      </param>
      <param>
        <value><str>[DEGREE]</str></value> <!-- <= 10 char -->
      </param>
      <param>
        <value><str>[DATE_OF_BIRTH]</str></value> <!-- YYYYMMDD Format -->
      </param>
      <param>
        <value><str>[STREET_ADDRESS]</str></value> <!-- <= 20 char -->
      </param>
      <param>
        <value><str>[OTHER_DESIGNATION]</str></value> <!-- <= 20 char -->
      </param>
      <param>
        <value><str>[CITY]</str></value> <!-- <= 25 char -->
      </param>
      <param>
        <value><str>[STATE]</str></value> <!-- <= 2 char -->
      </param>
      <param>
        <value><str>[ZIP]</str></value> <!-- <= 10 char -->
      </param>
      <param>
        <value><str>[COUNTRY]</str></value> <!-- <= 2 char -->
      </param>
    </params>
  </methodCall>
</xmlRPC>
```

Response

```
<?xml version="1.0"?>
<xmlRPC>
  <methodResponse>
    <params>
      <param>
        <value><str>[PATIENT_HASH]</str></value>
      </param>
    </params>
  </methodResponse>
</xmlRPC>
```

Discussion

interop.p_hash

Response

```
<?xml version="1.0"?>
<xmlRPC>
  <methodResponse>
    <params>
      <param>
        <value><str>[PATIENT_HASH]</str></value>
      </param>
    </params>
  </methodResponse>
</xmlRPC>

  <param>
    <value><str>[MIDDLE_INITIAL]</str></value> <!-- <= 1 char -->
  </param>
  <param>
    <value><str>[SUFFIX]</str></value> <!-- <= 10 char -->
  </param>
  <param>
    <value><str>[PREFIX]</str></value> <!-- <= 10 char -->
  </param>
  <param>
    <value><str>[DEGREE]</str></value> <!-- <= 10 char -->
  </param>
  <param>
    <value><str>[DATE_OF_BIRTH]</str></value> <!-- YYYYMMDD Format -->
  </param>
  <param>
    <value><str>[STREET_ADDRESS]</str></value> <!-- <= 20 char -->
  </param>
  <param>
    <value><str>[OTHER_DESIGNATION]</str></value> <!-- <= 20 char -->
  </param>
  <param>
    <value><str>[CITY]</str></value> <!-- <= 25 char -->
  </param>
  <param>
    <value><str>[STATE]</str></value> <!-- <= 2 char -->
  </param>
  <param>
    <value><str>[ZIP]</str></value> <!-- <= 10 char -->
  </param>
  <param>
    <value><str>[COUNTRY]</str></value> <!-- <= 2 char -->
  </param>
</params>
</methodCall>
</xmlRPC>
```

interop.get

Table of Contents ▲

- [interop.get](#)
- [Definition](#)
- [Request](#)
- [Response](#)

Definition

This function sends a patient hash to the server. A transaction ID is returned from the server as the request propagates to the logged in nodes.

Node transmits the hash of the requested patient to the server. The server returns a response ID. The server then queries all the logged in nodes, concatenates the data, and returns it along with the transaction id (in the node.accept function).

Request

```
<?xml version="1.0"?>
<xmlRPC>
  <methodCall>
    <methodName>get</methodName>
    <params>
      <param>
        <value><str>[PATIENT_HASH]</str></value>
      </param>
    </params>
  </methodCall>
</xmlRPC>
```

Response

```
<?xml version="1.0"?>
  <methodResponse>
    <params>
      <param>
        <value><str>[TRANS_ID]</str></value>
      </param>
    </params>
  </methodResponse>
</xmlRPC>
```

Discussion

node.status

Table of Contents ▲

- node.status
- Definition
- Request
- Response

Definition

This function allows the InterOp server to ping the node and has the nodes status code returned.

InterOp server sends 'PING' request to the node. The node then sends back a status along with a timestamp. If the node returns an error status, it will be logged out of the InterOp server. A null response assumes there are communication issues and the node would be logged out of the server.

Request

```
<?xml version="1.0"?>
<xmlRPC>
  <methodCall>
    <methodName>status</methodName>
    <params>
      <param>
        <value><string>PING</string></value>
      </param>
    </params>
  </methodCall>
</xmlRPC>
```

Response

```
<?xml version="1.0"?>
  <methodResponse>
    <params>
      <param>
        <value><str>[OK|DOWN|ERROR]</str></value>
      </param>
      <param>
        <value><dateTime.iso8601>YYYYMMDDTHHMMSS</dateTime.iso8601></value>
      </param>
    </params>
  </methodResponse>
</xmlRPC>
```

Discussion

node.accept

Table of Contents

- node.accept
- Definition
- Request
- Response

Definition

This function allows a node to accept the requested data from the InterOp server.

The server sends the data to the node along with the transaction ID and the MD5 Hash sum of the blob data. Upon receipt, the node generates its own MD5 Hash Sum on the transmitted data. Should the match, the transmission is a success and response of OK with timestamp is sent back to the server. Should they not match, an error state is returned to the server, which will retransmit at that time.

Request

```
<?xml version="1.0"?>
<xmlRPC>
  <methodCall>
    <methodName>accept</methodName>
    <params>
      <param>
        <value><str>[TRANS_ID]</str></value>
      </param>
      <param>
        <value><str>[P_DATA]</str></value>
      </param>
      <param>
        <value><str>[P_HASH]</str></value>
      </param>
    </params>
  </methodCall>
</xmlRPC>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value><str>OK|ERROR</str></value>
    </param>
    <param>
      <!-- If OK -->
      <value><str>[MD5 HASH SUM OF P_DATA]</str></value>
      <!-- If ERROR -->
      <value><str>[REASON FOR ERROR]</str></value>
    </param>
  </params>
</methodResponse>
</xmlRPC>
```

Discussion

node.get

Table of Contents ▲

- node.get
- Definition
- Request
- Response

Definition

This function returns data from a node to the InterOp server using a generated transaction ID.

The node returns data to the InterOp server. Data is associated to a transaction ID that is unique to the patient being requested and is not transmitted with the patient hash for security purposes. The MD5 Hash sum of the data is sent to the server for verification purposes of transmitted data.

Request

```
<?xml version="1.0"?>
<xmlRPC>
  <methodCall>
    <methodName>get</methodName>
    <params>
      <param>
        <value><str>[PATIENT_HASH]</str></value>
      </param>
    </params>
  </methodCall>
</xmlRPC>
```

Response

```
<?xml version="1.0"?>
  <methodResponse>
    <params>
      <param>
        <value><str>[PATIENT_DATA]</str></value>
      </param>
      <param>
        <value><str>[DATA_HASH]</str></value>
      </param>
    </params>
  </methodResponse>
</xmlRPC>
```

Discussion

Patient Hash Algorithm Overview

Table of Contents

- Patient Hash Algorithm Overview
- Preliminary Salted Hash Algorithm Definition
- Preliminary Salted Hash Algorithm Example

The server sends the data to the node along with the transaction ID and the MD5 Hash sum of the blob data. Upon receipt, the node generates its own MD5 Hash Sum on the transmitted data. Should the match, the transmission is a success and response of OK with timestamp is sent back to the server. Should they not match, an error state is returned to the server, which will retransmit at that time.

Preliminary Salted Hash Algorithm Definition

Based on HL7 2.3

Patient Field	Padded Field Length
Patient Given Name	25
Patient Family Name	25
Patient Middle Initial	1
Patient Suffix	10
Patient Prefix	10
Patient Degree	10
Patient DateOfBirth	8 (YYYYMMDD)
Street Address	20
Other Designation	20
City	25
State or Province	2
Zip or Postal Code	10
Country	2
Combined Size	175

String is combined using space padding to preserve field sizes

A 'salt' is added to prevent decryption. This is a secret string of characters that is prepended to the concatenation of fields. For this example we will use the phrase 'LITSinterOp' as the 'salt' (11 Char). In a production environment, the salt would be a randomized secret sting.

Total Combines Size of string: 186

Preliminary Salted Hash Algorithm Example

[End of padding represented by bracket]

```
[Test  
[Patient  
[A]  
[  
[  
[  
[19760717]  
[1234 AnyStreet Dr. ]  
[  
[Toledo ]  
[OH]  
[US]  
  
[Salt + Concatenation]
```

Hash Concatenation: [LITSinterOpPatient Test A 197607171234 AnyStreet Dr.]

Hash Sum: 6af3f2ed8331a2463f2f6de7da63e852

Discussion

1 Ian Ford: 2009/02/11 09:12

Is there any particular reason why the return values are defining type via the xsi instance namespace rather than relying on these being defined as string (or other specific types) in the service definition? Dynamically responding at runtime to type definitions is certainly necessary in certain situations, however it adds complications, and for the plotting phase where we are now it would be a useful simplification to define the types in the service spec.

Also - I tried the WSDL I included before - it gives an error on trying to deserialize the return values - I need to revise how the results array is described.

1 Ian Ford: 2009/02/10 11:44

Here's an initial attempt at creating a wsdl that covers the status method. It works successfully for requests submitted via Oxygen. Have not yet tested it for creating a language binding e.g. to Java using Axis.

```
<?xml version="1.0"?> <wsdl:definitions xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/soap/envelope/"
  targetNamespace="http://141.214.4.75/soap/"
  xmlns:ns1="http://141.214.4.75/soap/"
  xmlns:ns2="http://141.214.4.75/"
  </wsdl:definitions>

  <types>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:ns1="http://141.214.4.75/soap/"
      xmlns="http://141.214.4.75/soap/"
      targetNamespace="http://141.214.4.75/soap/" elementFormDefault="qualified">
      <xsd:element name="status">
        <xsd:complexType base="xsd:string" />
      </xsd:element>
    </xsd:schema>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:ns1="http://141.214.4.75/soap/"
      xmlns="http://141.214.4.75/" elementFormDefault="qualified">
      <xsd:element name="statusResponse">
        <xsd:complexType base="xsd:string" />
      </xsd:element>
    </xsd:schema>
  </types>

  <message name="statusRequest_m">
    <part name="status" element="ns1:status"/>
  </message>
  <message name="statusResponse_m">
    <part name="statusResponse" element="ns2:statusResponse"/>
  </message>

  <portType name="litsInterface">
    <operation name="run_status">
      <input message="s0:statusRequest_m"/>
      <output message="s0:statusResponse_m"/>
    </operation>
  </portType>

  <binding name="litsBinding" type="s0:litsInterface">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="run_status">
      <soap:operation soapAction="status" style="document"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>

  <service name="litsService">
    <port name="litsServiceSoap" binding="s0:litsBinding">
      <soap:address location="http://141.214.4.75/soap/" />
    </port>
  </service>

</wsdl:definitions>
```

1 Ian Ford: 2009/02/10 11:41

A couple of question on the namespace used in the SOAP requests and responses.

First a different namespace is used in the requests and responses. <http://141.214.4.75/soap/> is used in the request and <http://141.214.4.75/> is used in the response. It would probably more convenient if they were the same.

Also a namespace identifier which is not based on a machine IP address would be preferable. Maybe something like <http://lits.org/soap/litsinterop/>. It's not necessary for this to map to an actual webservice - it's just used as an identifier for the namespace.

Thanks Ian

1 Ian Ford: 2009/02/09 18:22

What is the URL for the SOAP service itself?

OK - I think I found it. <http://141.214.4.75/soap/>

That certainly gives a response to a status request

