A Web Service Monitoring System for the Enterprise

Coimbatore S. Chandersekaran
Institute for Defense Analyses, 4850 Mark Center Dr.
Alexandria, Virginia 22311 USA

and

William R. Simpson
Institute for Defense Analyses, 4850 Mark Center Dr.
Alexandria, Virginia 22311 USA

ABSTRACT

An enterprise that uses web services for the conduct of business can benefit from computer-based monitoring for its normal course of business. Services that are unavailable, delayed, inadequate, and/or provide poor or delayed information flow, all hinder or prevent the normal course of business. In extreme cases they may prevent business from being conducted. The proper performance of the service-oriented approach, the communication flow and the services themselves directly equate to the health and vitality of the enterprise. By health we mean, availability, performance, integrity, and reliability of web services. This paper reviews an agent based approach for web service monitoring in an enterprise environment. The agents create and collect information about the services. The paper also provides a definition of events that need to be monitored and the elements that should be recorded. Some information about critical events is time critical and should be sent as alerts to monitoring personnel for review and possible action. These processes are currently being implemented in a major defense enterprise.

Keywords: Help Desk, agent, monitoring, enterprise, support services, information sharing.

1. INTRODUCTION

Many organizations are standing up a high assurance, internet scale, web service based, enterprise system for information sharing. There will be many services in these systems and one service which is to be hosted is the help desk or the Enterprise Support Desk (ESD). In order to ensure a smooth and orderly disposition of all services within the enterprise, the ESD must be configured to quickly resolve problems within the infrastructure and hosted services, as well as external services that are used by the hosted services. In order to accomplish this, the help desk must have available information obtained from the Platform and Infrastructure Support Service Provider (PaaS and IaaS) as well as monitored information about the health state, security, and availability of hosted services. The ESD, as a hosted service is required to meet all aspects of the security in the high assurance enterprise, including but not limited to:

- Enterprise Naming;
- Credentialing of all active entities;
  - Enterprise based X.509 Certificates;
  - Provisioning of attributes associated with the Identity of the active entity for the creation of secondary credentials such as SAML;
- Bi-lateral end-to-end Public Key Infrastructure (PKI) authentication on all communication between active entities;
- Fully encrypted end-to-end transactions;
- And SAML based access control.

The publication of this paper does not indicate endorsement by the Department of Defense or IDA, nor should the contents be construed as reflecting the official position of these organizations.

The Enterprise Support Desk (ESD) is the combination of people, hardware, deployed software agents, and software displays, which maintain the health of the enterprise (Service-oriented architecture) SOA operations. It is both pro-active and re-active. It is required to be integrated with hardware and software health monitoring systems.

The ESD will consist of three levels:

1. Level 1 - customer support and help desk. A user may ask for assistance through a help desk phone number, e-mail, or web form entry. This is the unit that will develop a service ticket and see it through to completion. It is anticipated that 90% of all service desk calls will be handled within twenty minutes.
2. Level 2 – Proactive monitoring of services. This unit will monitor network activities and the performance of services using a testing tool. This unit will assist level 1 when resolution of help desk requests indicates it will help. It may also be used to generate service tickets before users call to report problems.
3. Level 3 – Active monitoring of security. This unit will monitor all security alerts and based upon heuristics developed within the enterprise will perform remote desk audits and dispatch a security monitoring team for physical audit of indicated desktops. This unit will assist level 1 when resolution of help desk requests indicates it will help.

This paper will deal with monitoring processes and required monitoring data. In order to facilitate all three units a series of agents on desktops and service machines, a knowledge base system, display work stations, and alert policies are needed. These requirements are varied and must be integrated with hardware and operating system health monitoring data. For this reason, all three units will use service monitoring software, which is flexible, configurable, expandable, and adaptable to include information from health state monitoring activities. Further, every desktop or laptop within the enterprise will have embedded agents for self help and repair of common problems as well as software (To Be Developed (TBD)) that will allow ESD principals to take control of the hardware for the purpose of auditing hardware and software configurations and provide troubleshooting assistance. Figure 1 shows the architecture of agents for the services management. There are separate agents for hardware health monitoring and two types of agents for each service. The first agent is embedded in the service itself for provision of alerts and internal logging of service data. The second agent is installed on the server and provides a sweep of log files, either periodically or on demand. At least one vendor (Amber Point [32]) provides both such agents.

Each Unit will have an administrator present with specific duties. Each shift will have such an administrator and the team of administrators will meet at a frequency dictated by events to review operations and to modify or create heuristics for ESD usage. Administrator privileges are required for certain tasks. All ESD personnel must abide by enterprise security policies, including bi-lateral authentication and SAML (Security Assertion Markup Language) [19, 36] authorization for access control. Figure 2 illustrates how these units work with the monitored data.

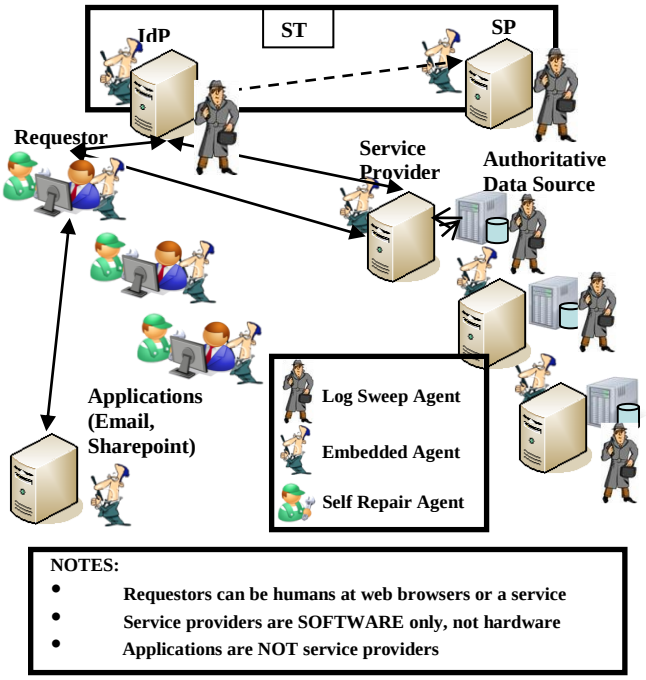

Figure 1 – Deployed Agent Architecture

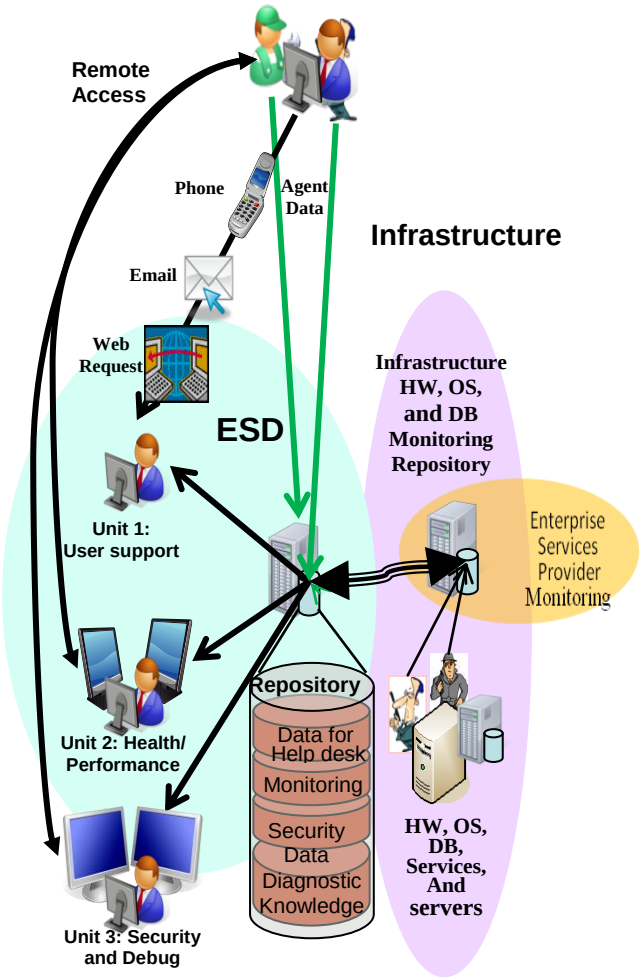

Figure 2 – Support Desk Operations

2. KNOWLEDGE BASE SYSTEM

The knowledge-base system is a single integrated source of all information on the operation of the enterprise. It will be updated by and accessible to all three units within the ESD. Instrumented agents gather records and feed the data base on a schedule or on demand. This subsection deals with information requirements for the Unit 2 ESD (SOA Monitoring)

Information for SOA Monitoring

The knowledge base is where all information related to the enterprise SOA is stored. This will include the following:

- Hardware/software current status from the infrastructure provider
- Current reports on test activities including response times, frequency of test, etc.
- Current reports of usage data from service agent monitors and service logs
- Hardware/software historical data
- A list of current alerts for the entire enterprise
- Historical data on alerts
- Configuration information

Within the enterprise there are three basic types of services:

- Web Applications
- Aggregation Services
- Exposure Services

The web application is setup to communicate with a users browser in HTML (Hypertext Markup Language). It may call aggregation services or exposure services. Each of these communicates using Simple Object Access Protocol (SOAP) /Extensible Markup Language (XML) [37]. For purposes of this paper the web application will be treated as an aggregation service.

Maintaining the Centralized Repository of Monitored Data

We will first examine the types of agents, the types of services in the enterprise, the requirements these services must meet, and then the monitoring data they must provide.

3. AGENTS IN THE ENTERPRISE

Agents are of three types, self help, embedded and monitor sweep. Each fulfills a different function in the architecture.

Self Help Agents

These agents are provided on the standard desktop and provide the user with a tool to examine configuration and software conflicts. They also allow support personnel at the enterprise helpdesk takeover of the desktop or device for diagnosis and repair of common software problems.

Embedded Agents

The embedded agent sits on the JAVA or .NET stacks and monitors the performance of the server and its threads. It should be configured to provide performance, connectivity and anomaly data to the log file for the service. It is unaware of sequence numbers or events transpiring within the service itself. It will be also configured to provide alert data as discussed below.

Monitor Sweep Agents

Figure 3 shows the placement of monitor sweep agents in the confines of the enterprise, even though they may reach outside the enterprise on external calls. It is the job of the monitor sweep agents to read, translate, and submit monitor records to the centralized data base. There will be no translation for custom developed services and translation will be minimal when Commercial Off-The-Shelf (COTS) products allow for Log4J[2] configurations. They may be extensive for other types of monitor data in external sources.

Data provided by the agents is insufficient to maintain the high availability and monitor security. The services themselves, who have the context for many events, must provide inputs to the monitoring data.

4. SERVICES BY TYPE

Within the enterprise there are three basic types of services:

- Web Applications
- Aggregation Services
- Exposure Services

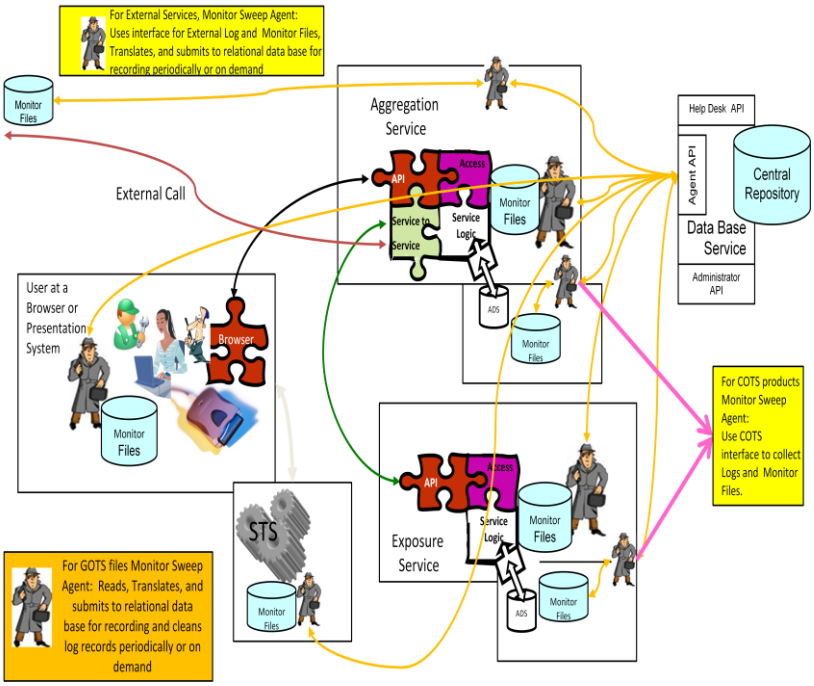

Figure 3 Web Service Monitoring Sweep Model

The web application is setup to communicate with a user's browser in HTML. It may call aggregation services or exposure services. Each of these communicates using SOAP/XML. For purposes of this paper the web application will be treated as an aggregation service.

[2] Log4J is a set of routines and formats provided by the Apache organization that provide for standard logging of records.

An aggregation service is defined as a service whose function is to provide data to the user from a number of authoritative data sources (ADS) using a series of exposure services or aggregation services and may include a number of widgets or other display code segments.

An exposure service is defined as a service whose primary function is to provide synthesized data to the user from one or more authoritative data sources, and this service may not call other exposure services, aggregation services but it may directly interface with one or more ADSs.

5. REQUIRED OF ALL SERVICES

The following are required of all services:

1. Monitoring Inputs - Once authentication and authorization are complete, dialogues should be monitored. All inputs in the dialogue should be reviewed for correctness, formatting, and other considerations before passing them on to data base or other code processors. Vulnerabilities occur with malformed inputs. All inputs should be read as general or strongly typed variables, tested for correctness and reformatted into the Application Program Interface (API) response when expected characteristics are verified. Malformed and nefarious inputs will be tested.

2. Credentials - Credentials are an integral part of the schema. Each service requiring access shall be credentialed by a trusted credentialing authority. Further, the Security token server that will be used for generating security assertion tokens must also be credentialed (primarily through the same credentialing authority, although others may be possible). For aggregation services, the credentialing includes not only registration with the Identity provider (initially restricted to Active Directory) but also the population of attributes with groups and roles necessary to access the exposure services. This can only be done by ESD administrators, so close coordination with them is required.

3. PKI required – X.509 Certificates - The primary exchange medium for setting up authentication of identities and setting up cryptographic flows is the PKI embodied in an X.509 certificate.

4. Bi-Lateral Authentication - The requestor will not only authenticate to the service (not the server), but the service will authenticate to the requestor. This two way authentication avoids a number of threat vulnerabilities. The requestor will initially authenticate to the server and set up an Secure Socket Layer (SSL) or Transport Layer Security (TLS) connection to begin communication with the service. The primary method of authentication will be through the use of public keys in the X.509 certificate, which can then be used to set up encrypted communications, (either by X.509 keys or a generated session key). The preferred method of communication is secure messaging, contained in SOAP envelopes. The difference between aggregation and exposure services is with aggregation service chaining the credential being passed to the services that are being called by the aggregation service, is the credential of the aggregation service not the original requestor. This is an

important distinction to make and developer of both aggregation and exposure services need to be aware of it. Because of the critical requirements associated with these packages and their consumption, the code to accomplish this is provided in JAR (or **J**ava[3] **AR**chive) files for use in implementations. Code developers are expected to use these code elements without modification.

5. Authorization Using Authorization Packages - All authorizations will be through the use of Authorization packages in accordance with either the Open Token Specification in Ping or the SAML 2.0 specification provided by OASIS [36]. Because of the critical requirements associated with authentication, the code to accomplish this is provided in JAR files for use in implementations. Code developers are expected to use these code elements without modification.

Under the current restrictions, all services developed will be either in a .NET environment or a J2EE environment. Each has specific requirements as noted below. Development of services are expected to take place in distinctive development cycles with the security aspects being handled last and close to the time of registration and certification.

6. SERVICE MODEL

The web application, aggregation service and exposure services are similar in their security and monitoring discussion as detailed in Figure 4.

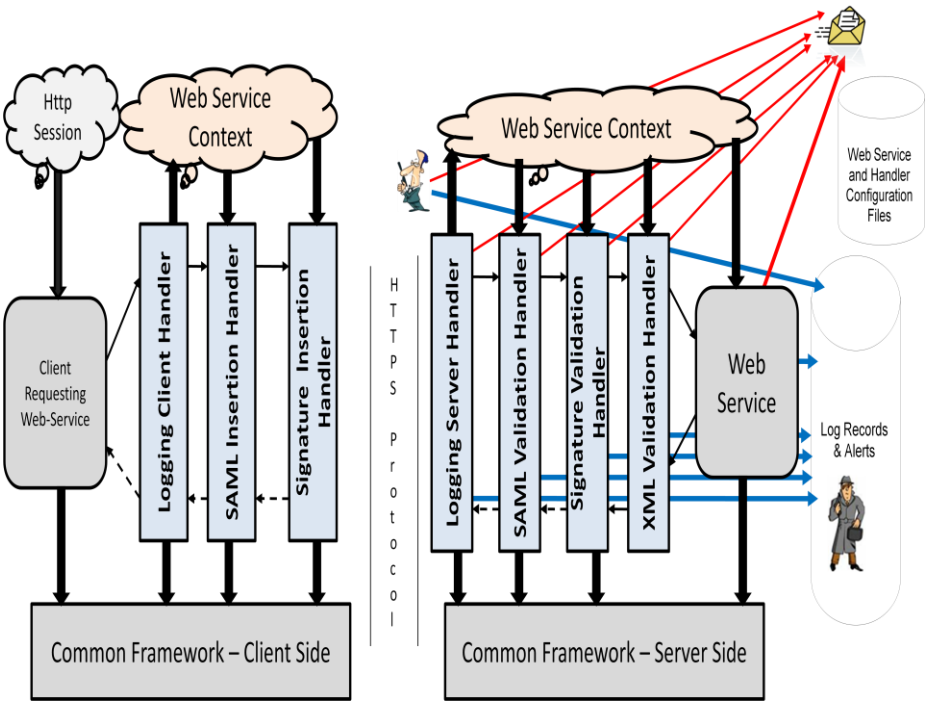

Figure 4 Web Service Context Logging/Monitoring Architecture

[3] Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible.

Anatomy of Web Services

The Web application, aggregation service or other web services will request data from one or more exposure services. It will compose, aggregate and synthesize data and either report its information to a web application, or formulate that data in XML for transfer to a web application or another service. This type of service has eight main parts shown in Figure 4. The code blocks are functional and not literal and the code may be developed in any fashion that meets the functional representation.

Code Elements of a Service

The figure shows an eight code block development used in the communication flow (from client requesting service to the web service) that will be used in infrastructure build-compatible web services software. The seven code rectangular blocks in light shading are provided by the enterprise. The code blocks are functional and not literal and the code may be developed in any fashion that meets the functional representation.

The eighth block is marked web service and contains the service logic, if any. This block is the responsibility of the developer. It contains an error handling routine for healing errors or posting monitor records and preparing return messages. It also contains an exit routine(s) for normal and abnormal exits. This routine creates monitor files and alerts as needed and formulates return messages when needed. This block is the responsibility of the developer.

Each service will have a management interface for setting the monitoring level as one of; Security, Basic, Enhanced. The levels are hierarchical in that Security records only those events marked security, Basic records those events marked Basic plus Security. Enhanced includes all events.

Monitor sweep agents reads, translates, and submits to relational data base for recording and cleans monitor records periodically or on demand. Monitor Sweep Agents may also request external monitor files, translates them, and submits them to relational data base for recording periodically or on demand.

Naming Schema

Each application will have a root name from which other names are derived. The root name should be reflective of the application and include within it a unique identification of the original applications for which the code was initially developed. Examples – Meta Data Environment (root name = MDE), Enterprise Rating System (root name = ERS). After some time, the number of applications may force the consultation of a registration list to avoid ambiguity.

Extended name will include the root name concatenated to the code name reflective of the service or usage of the code. Examples – ERS Web Application (ERSWebApp), MDE Search application service (MDESearch)

Configuration Files. As shown in Figures 3 and 4, there are various configuration files that must be maintained. These include:

1. Security Related addresses and data
2. Access Control Data
3. Authoritative Data Source location and data
4. Configuration files for Service use
5. Configuration files for the applications server +.

All of these files will be statically encrypted using the public key of the service to protect them from browsing and exfiltration[4]. Each will be described separately.

Security Related Addresses and Data - Table 1 provides the data elements necessary for this area:

Table 1 Security Related Addresses and Data

Element	Description	Comments
STS name	The registered name of the local STS	
STS address	URI of the local STS	
STS Public Key	256 bit key	For authentication
Service Name	The registered name of the service	
Service Address	URI of the service	
Service Public Key	256 bit key	For authentication
…		

The developer is responsible for populating this configuration file with data provided by the enterprise.

Access Control Data - Table 2 provides the data elements necessary for this area:

Table 2 Access Control Data

Element	Description	Comments
Attribute 1	An attribute that allows access	For authorization
Attribute 2	An attribute that allows access	For authorization
…	May contain up to 512 attributes	For authorization
Attribute a	An attribute that denies access	For denial of authorization
Attribute b	An attribute that denies access	For denial of authorization
…	May contain up to 512 attributes	For denial of authorization

The developer is responsible for providing an administrative interface to the service for maintenance of this data.

Authoritative Data Sources - Table 3 provides the data elements necessary for this area:

Table 3 Authoritative Data Sources

Data Element	Description
AD1 name	The registered name of the first

[4] Exfiltration is an antonym for infiltration, similar to extraction.

Table 3 Authoritative Data Sources

Data Element	Description
	Authoritative Data Source
AD1 address	URI of AD1
AD1 Public Key	256 bit key
AD2 Name	The registered name of the second Authoritative Data Source
AD2 Address	URI of AD2
AD2 Public Key	256 bit key
…	May contain as many as needed

The developer is responsible for populating this configuration file with data provided by the enterprise. Population with dummy data will be done until the final authority to operate, and registration of the service is accomplished.

Configuration files for Service Use - Configuration files for use by the service may be in developer format, but should contain data consistent with indirect addressing and reference so that when things in the enterprise change or move; they become a configuration issue in lieu of a service re-write issue.

Configuration files - applications server and others - These files are application specific and must be populated as necessary.

Audit Activities - Security audits involve several classes or audit issue areas including: data generation, automatic response, records analysis, records review, event selection, record storage.

7. DATA STORAGE ARCHITECTURE

Each application will have a root name from which other names are derived. The root name should be reflective of the application and include within it a unique identification of the original applications for which the code was initially developed.

Monitor Files

Monitor files will be stored in application storage files similarly to configuration files as follows:
Activity logs as defined below:
{Storage root}: /{rootname}files/{extendedname}Monitor/{extendedname}Monitor.log
Ex:C:/ERSfiles/ERSWebAppMonitor/ERSWebAppMonitor.log
An agent present on the server will either periodically or on demand be configured to sweep the log file and send it to the enterprise data base for storage and later analysis.

Developer Debug Files

Debug and other messages added to the required monitoring files are permitted and will be sent to the file:
{Storage root}:/{root name}files/{extendedname}Monitor/{extendedname}MonitorSUP/ {extendedname}MonitorSUP.log
Ex: C:/ERSfiles/ERSWebAppMonitor/ERSWebAppMonitorSUP/ ERSWebAppMonitorSUP.log

Debug files will follow the Log4J format provided in Table 4 and Annex A. An agent present on the server will either

periodically or on demand be configured to sweep the monitor file and send it to the enterprise data base [or to a data base of the developer's choice] for storage and later analysis.

Data Generation

This activity defines requirements for recording the occurrence of relevant events that take place. The security level of recording is the default and must be recorded in the file name described above. It may be supplemented by the basic level of data recording as ordered by the enterprise support desk. The Basic is selectable through administrative API with administrator privilege. The activity identifies the level of monitoring, enumerates the types of events that shall be monitorable by the service (except as noted – some data may come from agents), and identifies the minimum set of monitor-related information that should be provided within various monitor record types. The Enhanced is selectable through administrative API with administrator privilege. This Enhanced level is selectable and is automatically triggered with any security alert (Table 5). For enterprise systems, monitor records are required for all security relevant events. Basic monitoring will add those events labeled as Basic to the security events. Enhanced monitoring will record all events in Table 4. A Sequence Number which is a 16 digit alpha numeric that identifies the overall transaction stream uniquely (Sn). Sn is given by:

Concatenate:
ip address of application server +
URI of service operation +
timestamp[yyyyMMddHHmmssSSS] +
thread ID +
Common Name of initial service requester

Any error messages returned to a requester inviting them to call the help desk would refer to the Common Name + URI + time approximation yyyyMMddHHmm.

If Sequence Number.eq.Null) then Sequence Number = ip address//'$.'//URI//'$.'//timestamp//'$.'//thread ID//'$.'//Common Name

Table 4 provides the events that require monitor records for this area:

Note:
Let Tn = Thread Number
Let Sn = Sequence Number
Let AN = Active Entity Name
Let EV = Event Name
Let ID = ID of Service Requester
Let IDr =ID of Service Requested
Let d = date/time in [yyyy-MM-dd-hh-mm-ss-"."ff(5digits)] (26 characters)
Let OP1 = Other Pertinent Data 1
Let OP2 = Other Pertinent Data 2
Let OP3 = Other Pertinent Data 3
Let OP4 = Other Pertinent Data 4
Let OP5 = Other Pertinent Data 5
Let EM = Exception Message
Let Gp = Group or role for Access
Let CDN = Cert Domain Name)
Let CPK = Cert Public Key
Let CAPII = Complete API input
Let CAPIR = Complete API Request
Let CAPIr = Complete API Return
Let SAML = Complete SAML Token
Let OPEN = Complete Open Token
Let IDR = ID of service requested
Let Rf= Reason for Authorization Failure
Enumeration (Expired SAML, Tampered SAML, Expired Open Token, Timeout, No Matching Groups, Unrecognized SAML, Unrecognized Open Token, Other)
Let Rfi = Reason for failed Session Initiation
Enumeration (Timeout, Server error, Corrupted Input, Unknown, Other)
Let Rfe = Reason for Premature Shutdown
Enumeration (Timeout, Hardware Issue, Server error, Corrupted Input, Unknown, Other)
Let SV = Other Security Violation
Enumeration (Timeout, Luna Hardware Issue, Server error, Corrupted Input, Unknown, Other)
Let Ta = Monitor Storage Threshold Exceeded
Enumeration (80%, 90%, 100%)
Let XMLDSig = XML Digital Signature block

Table 4 Audit Records

Event/Category	Content
Start-up of the monitor functions within the service Monitor Levels: • None • Alerts and Inputs • Trace **Category = Basic** **Responsibility = Embedded Agent**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, Other pertinent data, XMLDSig[5]
	Event Name = " Start Up Monitor" OP1= ID of Service Requester OP2 = Level of Monitor OP3- 5 = {developer use}
Change in monitor functions within the service **Category = Basic** **Responsibility = Web Service**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, Other pertinent data, XMLDSig
	Event Name = "Shut Down Monitor" OP1= ID of Service Requester OP2 = Level of Monitor OP3-5 = {developer use}
Session initiation activities **Category = Basic** **Responsibility = Web Service**	Thread Number, Sequence Number, Active Entity Name, ID of Service Requester, Event Name, Date/ time, Other pertinent data, XMLDSig
	Event Name = "Session Initiated" OP1= ID of Service Requester OP2 - 5 = {developer use}
Input malformed **Category = Basic**[6] **Responsibility = Web Service**	Thread Number, Sequence Number, Active Entity Name, ID of Service Requester, Event Name, Date/ time, Other pertinent data, XMLDSig
	Event Name = "Malformed Input" OP1= ID of Service Requester OP2 = input that triggered the error OP3-5 = {developer use}
Output malformed **Category = Basic**[5] **Responsibility = Web Service**	Thread Number, Sequence Number, Active Entity Name, ID of Service Requester, Event Name, Date/ time, Other pertinent data, XMLDSig
	Event Name = "Malformed Output" OP1= ID of Service Requester OP2 = output that triggered the error

[5] The XMLDSig is for tamper protection and may be inserted by the service or the logging routine using he private key and certificate of the service.
[6] The developer requirements for these errors are covered in a separate section following alert data.

Table 4 Audit Records

Event/Category	Content
	OP3-5 = {developer use}
Session complete **Category = Basic** **Responsibility = Web Service**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, Other pertinent data, XMLDSig
	Event Name = "Session Complete" OP1= ID of Service Requester OP2-5 = {developer use}
Any exceptions due to Java or supporting programs **Category = Basic**[5] **Responsibility = Web Service and all Handlers as well as web server**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, Other pertinent data, XMLDSig
	Event Name = "Exception" OP1= ID of Service Requester OP2 = Exception Message OP3 = data specific to exception or cure OP4 = data specific to exception or cure OP5 = data specific to exception or cure Return to Next level Server Problem – call help desk (Sequence Number =SN) {Unless self healing} or Server Problem – retry (Common Name + URI + time approximation yyyyMMddHHmm.) {Unless self healing}
Any known violations of security policy, including all instances of alerts in Table 5 and others as needed. **Category = Security** **Responsibility = Web Service and all Handlers as well as web server**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, Other pertinent data, XMLDSig
	Event Name = "Security Violation" OP1= ID of Service Requester OP2 = {developer use} OP3 = {developer use} OP4 = {developer use} OP5 = {developer use} Return to Next level Server Problem – call help desk (Common Name + URI + time approximation yyyyMMddHHmm.)
Authorization **Category = Enhanced** **Responsibility = SAML Validation Handler**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, SAML Token, Open Token, other pertinent data, XMLDSig
	Event Name = "Authorization" OP1 = ID of Service Requester OP2 = SAML OP3 = Open Token OP4-5 = {developer use}
Failed authentication **Category = Security** **Responsibility = web server**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, User Entity identity, Other pertinent data, XMLDSig
	OP1 = ID of Service Requester OP2 = Requester Cert DN OP3 = Requester Cert Public Key OP4 = Requester Cert Revocation Date OP5 = Result of Cert Validation Check Return to Next level Server Problem – call help desk (Common Name + URI + time approximation yyyyMMddHHmm.)
Authentication **Category = Enhanced** **Responsibility = web server**	Thread Number, Sequence Number, Active Entity Name, Event Type, ID of Service Requester, Date/time, other pertinent data, XMLDSig

Table 4 Audit Records

Event/Category	Content
	Event Name = Authentication OP1= ID of Service Requester OP2 = Requester Cert DN OP3 = Requester Cert Public Key OP4 = Requester Cert Revocation Date OP5 = Result of Cert Validation Check
Failed authorization **Category = Security** **Responsibility = SAML Validation Handler**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, User Entity identity, Other pertinent data, XMLDSig
	OP1 = ID of Service Requester OP2 = SAML OP3 = Open Token OP4 = {developer use} OP5 = {developer use} Return to Next level Server Problem – call help desk (Common Name + URI + time approximation yyyyMMddHHmm.)
Thread Establishment[7] **Category = Enhanced Responsibility = Embedded Agent**	Thread Number, Sequence Number[11], Active Entity Name, Event Type, ID of Service Requester, Date/time, Complete API input, other pertinent data, XMLDSig
	Event Name = Thread Established OP1= ID of Service Requester Op2 = Complete API Input OP3 = {agent developer use} OP4 = {agent developer use} OP5 = {agent developer use}
Service Request – Each **Category = Enhanced** **Responsibility = Web Service**	Thread Number, Sequence Number, Active Entity Name, Event Type, ID of Service Requested, Date/time, Complete API request, other pertinent data, XMLDSig
	Event Name = Service Request OP1= ID of Service Requester Op2 = Complete API Request OP3-5 = {developer use)
Service Response – each If the thread, and sequence as well as the ID of the service are the same the wait time is the time difference between this and the previous record **Category = Enhanced Responsibility = Web Service**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requested, Date/time, Complete API return other pertinent data, XMLDSig
	Event Name = "Service Response" OP1= ID of the Service Requester Op2 = Complete API return OP3 -5 = {developer use}
Content Download[8] – used when notify or other messages are sent to requester. **Category = Basic** **Responsibility = Web Service**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, Complete API return other pertinent data, XMLDSig
	Event Name = "Content Download" OP1= Content Name Op2 = Unique ID, if one assigned OP3-5 = {developer use)

[7] The java agent must trap thread establishment, or the server must report thread establishment.

[8] Content objects here are meant to include doc, ppt, pdf, jpeg, etc. and not the normal data transfers associated with authoritative data bases.

Table 4 Audit Records

Event/Category	Content
Message Acknowledgement – used when notify or other messages are sent to requester. May be MAC notices, download restrictions, etc. **Category = Basic Responsibility = Web Service**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, Complete API return other pertinent data, XMLDSig
	Event Name = "Message Acknowledgement" OP1= Acknowledgement Message Op2 = Acknowledgement Response (Yes, No, etc-may only be one (ok) OP3-5 = {developer use}
Health State Performance Monitors	
Final Rollout Performance Data A program segment could be computational or waiting for inputs, etc. **Category = Enhanced Responsibility = Web Service**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, Other pertinent data, XMLDSig
	Event Name = "Performance" OPi= Program Segment Name i plus delta ti (i=1-5)
Timeouts and work around Timeouts may or may not occur **Category = Enhanced Responsibility = Web Service**	Thread Number, Sequence Number, Active Entity Name, Event Name, , ID of Service Requester, Date/time, Other pertinent data, XMLDSig
	Event Name = "Timeouts" OP1 = Timeout occurrence location OP2 = Timeout delta t OP3 = Workaround (e.g., cache data, default value, etc.) OP4 -5= {developer use}
Developer Debug Records	
Developer Debug Records[9] **Category = Enhanced Responsibility = Web Service - optional**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, Other pertinent data, XMLDSig
	Event Name = [Name Provided by Developer] OP1 -5= {developer use}

In addition, the service developer must meet any requirements of the ESD.[10] The developer will determine the content of the enhanced and verbose log providing information necessary for debugging and forensics that apply to the service itself.

Alerts and Automatic Responses - Certain activities that may happen are considered security violations and will require the computer or network to send alerts and automatically respond. Activities such as failed authentication, attempts to access data beyond access authority and the potential loss of data due to resource constraints are typical examples. Automated responses may include, among other things, session termination, suspension of privileges and in extreme cases, shutting down computers and other equipment. Table 5 provides the events that require alerting and automatic response.

Table 5 Alert Data

Event	Alert Content	Response
Failed authentication	Thread Number, Sequence Number, Active	Server Problem – call help desk

Table 5 Alert Data

Event	Alert Content	Response
Responsibility = web server	Entity Name, Event Name, ID of Service Requester, Date/time, User Entity identity, Other pertinent data	(Common Name + URI + time approximation yyyyMMddHH mm.)

Message Body[11]
`Subject: Alert message-Failed Authentication`
`OP1 = ID of Service Requester`
`OP2 = Requester Cert DN`
`OP3 = Requester Cert Public Key`
`OP4 = Requester Cert Revocation Date`
`OP5 = Result of Cert Validation Check`
`XMLDSig`		

Failed authorization **Responsibility = SAML Validation Handler**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, Reason for failure of the event, Other pertinent data	Server Problem – call help desk (Common Name + URI + time approximation yyyyMMddHH mm.)

`Subject: Alert message-Failed Authorization`
`OP1 = ID of Service Requester`
`OP2 = SAML`
`OP3 = Open Token (if used)`
`OP4-5 = {developer use}`
`XMLDSig`		

Failed session initiation activities **Responsibility = whoever has visibility**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, Reason for failure of the event, Other pertinent data	Server Problem – call help desk (Common Name + URI + time approximation yyyyMMddHH mm.)

`Subject: Alert message-Failed Session`
`OP1= ID of Service Requester`
`Op2 = Complete API Input`
`OP3 = Indicated data`
`OP4-5 = {developer use}`
`XMLDSig`		

Abnormal or Premature Session Shutdown[12] **Responsibility = Agent Produced**	Thread Number, Sequence Number[13], Active Entity Name, Event Name, ID of Service Requester, Date/time, Reason for shutdown of the event, Other pertinent data	Server Problem – call help desk (Common Name + URI + time approximation yyyyMMddHH mm.)

`Subject: Alert message-Session Shutdown`
`OP1 = ID of Service Requester`
`OP2 = Reason for Shutdowm`
`OP3 = Indicated data`
`OP4-5 = {agent developer use}`
`XMLDSig`		

Any known violations of security policy not otherwise listed here. **Responsibility = Web Service and all Handlers as well as web server**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, Violation, Other pertinent data	Server Problem – call help desk (Common Name + URI + time approximation yyyyMMddHH mm.)

`Subject: Alert message-Security Violation`
`OP1 = ID of Service Requester`
`OP2 = Security Violation`
`OP3-5 = {developer use}`
`XMLDSig`		

[9] Developer debug files are optional.
[10] See Life Cycle Management of Service, current version SAF/XCT. Most recent version at this writing is 1/28/2009 Version 0.955.

[11] SMTP see below – all alerts use the same message format
[12] If the error routine does not trap the need for shut down then the java agent must trap it on the thread, or the server must report it.
[13] May not be available.

Table 5 Alert Data

Event	Alert Content	Response
Monitor Data Full[14] (Indicated by file write error) **Responsibility = Agent Produced**	Thread Number, Sequence Number[11], Active Entity Name, Event Name, ID of Service Requester, Date/time, Other pertinent data	Activity shutdown

`Subject: Alert message-Monitor Data Full`
`OP1 = ID of Service Requester`
`OP2 = Ta`
`OP3 = Indicated data`
`OP4-5 = {agent developer use}`
`XMLDSig`		

Record has been tampered with[15] **Responsibility = Monitor Agent or data base produced**	Thread Number, Sequence Number[11], Active Entity Name, Event Name, ID of Service Requester, Date/time, Missing Monitor Record, Other pertinent data	Server Problem – call help desk (Common Name + URI + time approximation yyyyMMddHH mm.)

`Subject: Alert message-Record Tampering`
`OP1 = ID of Service Requester`
`OP2 = Missing numbers`
`OP3 = Indicated data`
`OP4-5 = {agent developer use}`
`XMLDSig`		

Requirements for Java and Service Exception Errors

The developer is required to provide a number of assurance processes before deploying a service in the operational environment.

Requirement 1. The developer must do a static code analysis and source code scan for vulnerabilities that is based on the CVE[16]. This report is to be delivered to the government with mitigation and rationale for all identified vulnerabilities.

Requirement 2. The developer must scan all inputs and outputs for "nominal" and report malformed inputs or outputs. The malformed information may be corrected or rejected before acting upon any information transmitted.

Requirement 3. The developer must provide a list of all java runtime exception errors that will be relevant, and provide rationale for any java runtime exception errors not on the list.

Requirement 4. Each relevant error must be documented as to how it is handled, including self-healing errors, and how each non-self-healing error will be handled within the service and how its reports should be handled in after action analyses.

Requirement 5. Each relevant java exception error must be trapped, and logged. Errors that terminate a session must increase the monitoring category to **Enhanced** until reset.

Record Storage

Requirements for the service to supply a clear audit trail.

Requirement 1. Protected audit trail storage requirements are placed on the audit trail. It will be protected from unauthorized deletion and/or modification. Authorized deletion or modification is logged.

Requirement 2. Guarantees of audit data availability specifies the guarantees that the system maintains over the audit data given the occurrence of an undesired condition. This requires resource planning and the enterprise generally requires 3 months online data and 2 years archived data.

Requirement 3. Action in case of possible audit data loss specifies actions to be taken if a threshold on the audit trail is exceeded. This will include appropriate notifications under automated response when thresholds are approached.

Requirement 4. Prevention of audit data loss specifies actions in case the audit trail is full. This should not happen, but the response should be to preserve the most recent records and to shut down non-essential activity until audit capability is restored.

8. SUMMARY

This paper has presented an agent-based monitoring system and data requirements for a web-service based enterprise. Although agent architectures have been used in the past [21, 22], we believe this is the first time that such an architecture has been fully integrated with monitoring information requirements in the services themselves. Additional elements of the architecture are provided in [44-48].

9. REFERENCES

[1]. Air Force Information Assurance Strategy Team, Air Force Information Assurance Enterprise Architecture, Version 1.25, SAF/XC, 11 April 2008.

[2]. AFPD 33-3 Information Management, AF Portal Community of Practice: AF Information & Data Management Strategy – Implementation (Policy) http://www.e-publishing.af.mil/

[3]. COI Primer, AF Portal Community of Practice: AF Information & Data Management Strategy – Implementation (COI Primer)

[4]. DoD Directive 8320.2 "Data Sharing in a Net-Centric Department of Defense" and DOD Guidance 8320.2-G "Guidance for Implementing Net-Centric Data Sharing", AF Portal Community of Practice: AF Information & Data Management Strategy – Implementation (Policy)

[5]. Metadata Concept, AF Portal Community of Practice: AF Information & Data Management Strategy – Implementation (Metadata)

[6]. Transparency Integrated Product Team (TIPT) information and proceedings AF Portal Community of Practice

[7]. AFI 33-115, Network Management and Licensing Network Users and Certifying Network Professionals

[8]. AFMAN 33-223, Identification and Authentication

[9]. AFMC Supplement 1, AFMAN 33-223, Identification and Authentication

[10]. CJCSI 3170.01E, Joint Capabilities Integration and Development System

[14] This data may not be available, and if it is will come from the Log4J routine. Probably from a write error if disk full error does not happen.

[15] This record must come from the signature check when the agent uploads the record to the data base – not required of service developer.

[16] Common Vulnerabilities and Exposures (CVE) is a list or dictionary that provides common names for publicly known information security vulnerabilities, http://cve.mitre.org/

[11]. CJCSI 6212.01D, Interoperability and Supportability of Information Technology and National Security Systems

[12]. DoDD 5000.1, The Defense Acquisition System

[13]. DoDD 4630.5, Interoperability and Supportability of Information Technology and National Security Systems

[14]. DoDD 8000.1, Management of DoD Information Resources and Information Technology

[15]. DoDD 8115.1, Information Technology Portfolio Management

[16]. DoDI 5000.2, Operation of the Defense Acquisition System

[17]. DoDI 8115.02, "Information Technology Portfolio Management Implementation", October 30, 2006

[18]. Joint Concept of Operations for Global Information Grid NetOps, Version 3, August 4, 2006

[19]. "Guide to Secure Web Services: Recommendations of the National Institute of Standards and Technology", NIST-US Department of Commerce Publication, August 2007.

[20]. Middleton, I "Key Factors in HelpDesk Success (An analysis of areas critical to helpdesk development and functionality.)" British Library R&D Report 6247, The British Library 1996

[21]. Tivoli Web Services Manager V1.7 Includes Tivoli Application Performance Management; Tivoli Web, Services Analyzer V1.7 Includes National Language Support, Software Announcement, December 4, 2001

[22]. Service management solutions White paper Deliver service excellence through the unique advantages of IBM Service Management solutions, September 2007.

[23]. J. A. Farrell and H. Kreger, Web services management approaches, IBM Systems Journal, Vol. 41, No. 2, 2002.

[24]. Managing Information Access to an Enterprise, Information System Using J2EE and Services, Oriented Architecture, IBM Redbook, January 2005.

[25]. WebSphere Application Server V6.1 Security, Handbook, IBM Redbook, December 2006.

[26]. Cool Vendors in SOA Governance, Gartner Group, 2008

[27]. Criteria for Evaluating a Vendor's SOA Governance Strategy, Gartner Group, May 2008

[28]. Criteria for Integrated SOA Governance Technology Sets, Gartner Group, January 2008

[29]. Key Issues for SOA Governance Technologies, Gartner Group, 2008

[30]. The Forrester Wave™: SOA Service Life-Cycle Management, Q1 2008, Fulton, Larry, January 28, 2008

[31]. SOA Governance Infrastructure, Maines, Ann-Thomas, Burton Group, November, 2007

[32]. Introduction to Amber Point SOA Management System, Software Manual, December 2007

[33]. Oracle Monitoring Tools, Oracle Data Sheet

[34]. Evaluating IBM, Microsoft, Oracle and SAP Commitment to SOA Governance, Gartner Group, October 07

[35]. Oracle Web Service Manager 10g3 Overview, http://www.oracle.com/technology/products/webservices_manager/htdocs/owsm_10gr3_fov_1.html

[36]. World Wide Web Consortium (W3C):

a. "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)", April 27, 2007.
b. W3C XML Schema Definition Language (XSD) 1.1 Part 1&2, 21 July 2011.
c. XML Encryption Requirements, 04 March 2002
d. XML Signature Syntax and Processing, 10 June 2008.
e. Web Services Description Language (WSDL) Version 2.0 Part 0-2, 26 June 2007.
f. Semantic Annotations for WSDL and XML Schema, 28 August 2007.
g. Web Services Architecture Requirements, 11 February 2004.
h. XHTML™ 1.1 - Module-based XHTML, 23 November 2010.

[37]. OASIS open set of Standards

a. N. Ragouzis et al., *Security Assertion Markup Language (SAML) V2.0 Technical Overview*, March 2008.
b. P. Madsen et al., *SAML V2.0 Executive Overview*, April 2005.
c. P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005.
d. S. Cantor et al. *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005.
e. S. Cantor et al. *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005.
f. S. Cantor et al. *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005.
g. F. Hirsch et al. *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005
h. J. Hodges et al. *Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005.

[38]. Standard for Naming Active Entities on DoD IT Networks, Version 3.5, Sept. 23, 2010.

[39]. Federal Information Processing Standards Publication, FIPS PUB 140-2, Security Requirements for Cryptographic Modules, National Institute of Standards and Technology, May 25, 2001.

[40]. Common Criteria for Information Technology Security Evaluation, Version 3.1, revision 3, July 2009.

[41]. Internet Engineering Task Force (IETF) Standards:

a. STD 66 (RFC3986) Uniform Resource Identifier (URI): Generic Syntax, T. Berners-Lee, R. Fielding, L. Masinter, January 2005
b. STD 9 (RFC0959) File Transfer Protocol, J. Postel, J. Reynolds, October 1985.
c. STD 5 (RFC0791) Internet Protocol, J. Postel, September 1981.
d. RFC 2459, Internet X.509 Public Key Infrastructure Certificate and CRL Profile, January 1999.

e. *LDAPv3: Technical Specification*, **RFC 3377**, September 2002. *Authentication Methods for LDAP*. M. Wahl, H. Alvestrand, J. Hodges, R.L. Morgan. RFC 2829 May 2000.

[42]. Naming and Addressing: URIs, URLs, ..., Dan Connolly, Revision: 1.58 of 2006/02/27, Created 1993 by TimBL http://www.w3.org/Addressing/

[43]. Formally Assigned Uniform Resource Names (URN) Namespaces, Last Updated 2011-08-17, This is the Official IANA Registry of URN Namespaces, http://www.iana.org/assignments/urn-namespaces/urn-namespaces.xml

[44]. William R. Simpson, Coimbatore Chandersekaran and Andrew Trice, The 1st International Multi-Conf. on Eng. and Tech. Innovation, *"Cross-Domain Solutions in an Era of Information Sharing"*, Volume I, pp.313-318, *Orlando, FL., June 2008.*

[45]. Coimbatore Chandersekaran and William R. Simpson, World Wide Web Consortium (W3C) Workshop on Security Models for Device APIs, *"The Case for Bi-lateral End-to-End Strong Authentication"*, 4 pp., London, England, December 2008.

[46]. William R. Simpson and Coimbatore Chandersekaran, 2nd International Multi-Conference on Engineering and Technological Innovation, Volume I, pp. 300-305, *"Information Sharing and Federation", Orlando, FL., July* 2009.

[47]. William R. Simpson and Coimbatore Chandersekaran, 1st International Conference on Design, User Experience, and Usability, part of the 14th International Conference on Human-Computer Interaction (HCII 2011), *"A Multi-Tiered Approach to Enterprise Support Services"*, 10pp, Orlando, FL., July 2011.

[48]. William R. Simpson, Coimbatore Chandersekaran and Ryan Wagner, Lecture Notes in Engineering and Computer Science, Proceedings World Congress on Engineering and Computer Science 2011, Volume I, *"High Assurance Challenges for Cloud Computing"*, pp. 61-66, San Francisco, October 2011.

Annex A

Log4J Handler File

Let Tn = Thread Number
Let Sn = Sequence Number
Let AN = Active Entity Name
Let EV = Event Name
Let ID = ID of Service Requester
Let d = date/time in [yyyy-MM-dd-hh-mm-ss-"."ff(5digits)
Let TZ=Time Zone 3 char
Let OP1= Other Pertinent Data 1
Let OP2= Other Pertinent Data 2
Let OP3= Other Pertinent Data 3
Let OP4= Other Pertinent Data 4
Let OP5= Other Pertinent Data 5
Let Dl = Delimiters ['!!']
Let XMLDSig = the XML Digital Signature Block

From Table 4 Lines1&2

Start-up of the monitor functions within the service Monitor Levels: • None • Alerts and Inputs • Trace **Category = Basic**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, Other pertinent data, XMLDSig
	Event Name = " Start Up Monitor" OP1= ID of Service Requester OP2 = Level of Monitor OP3 = {developer use} OP4 = {developer use} OP5 = {developer use}
Change in monitor functions within the service **Category = Basic**	Thread Number, Sequence Number, Active Entity Name, Event Name, ID of Service Requester, Date/time, Other pertinent data, XMLDSig
	Event Name = "Shut Down Monitor" OP1= ID of Service Requester OP2 = Level of Monitor OP3 = {developer use} OP4 = {developer use} OP5 = {developer use}

For both
If M= printf (%AN%Dl %EV%Dl %OP1%Dl %OP2 %Dl %OP3%Dl %OP4%Dl %OP5%Dl %XMLDSig)
Then
M transfers to the logging server handler.

Annex B

SMTP File Format for Alerts

The service, at boot-up should be configured to open a mutually authenticated SSL session with the help desk mail processor.

Let Tn = Thread Number
Let Sn = Sequence Number
Let AN = Active Entity Name
Let EV = Event Name
Let ID = ID of Service Requester
Let d = date/time in [yyyy-MM-dd-hh-mm-ss-“.”ff(5digits)]
Let OP1= Other Pertinent Data 1
Let OP2= Other Pertinent Data 2
Let OP3= Other Pertinent Data 3
Let OP4= Other Pertinent Data 4
Let OP5= Other Pertinent Data 5
Let XMLDSig = the XML Digital Signature Block

Failed authentication	Thread Number, Sequence Number, Active Entity Name, Event Name , ID of Service Requester, Date/time, Type of event, User Entity identity, Service Entity Identity, Other pertinent data, XMLDSig	Server Problem – call help desk (a Sequence Number may be provided) to user

SMTP transport

The format for sending a message via SMTP to one mailboxes (*alertbox* located in the helpdesk mail domain (*helpdesk.com*) is reproduced in the following session exchange.

For illustration purposes here (not part of protocol), the protocol exchanges are prefixed for the server (*S:*) and the client (*C:*).

After the message sender (SMTP client) establishes a reliable communications channel to the message receiver (SMTP server), the session is opened with a greeting by the server, usually containing its fully qualified domain name (FQDN), in this case *smtp.example.com*. The client initiates its dialog by responding with a HELO command identifying itself in the command's parameter with its FQDN (or an address literal if none is available).

Message Body

S: 220 smtp.helpdesk.com ESMTP Postfix
C: HELO relay.helpdesk.org
S: 250 Hello relay.helpdesk.org, I am glad to meet you
C: MAIL FROM:<AN>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: AN <AN>
C: To: "alertbox" <alertbox@helpdesk.com>
C: Date: DN
C: Subject: Alert message - EV
C:
C: Let Record Number = Rn
C: Thread Number = Tn
C: Sequence Number = Sn
C: Active Entity Name = AN
C: Event Name = EV
C: d
C: OP1
C: OP2
C: OP3
C: OP4
C: OP5
C: XMLDSig
C: .
S: 250 Ok: queued as 12345

The client notifies the receiver of the originating email address of the message in a MAIL FROM command. Successful reception and execution of a command is acknowledged by the server with a result code and response message (e.g., 250 Ok).

The transmission of the body of the mail message is initiated with a DATA command after which it is transmitted verbatim line by line and is terminated with an end-of-data sequence. This consists of a new-line (<CR><LF>), a single full stop (period), followed by another new-line.

The server's positive reply to the end-of-data, implies that the server has taken the responsibility of delivering the message. A message can be doubled if there is a communication failure at this time, e.g. due to a power shortage: Until the sender has not received that 250 reply, it must assume the message was not delivered. On the other hand, after the receiver has decided to accept the message, it must assume the message has been delivered to it. Thus, during this time span, both agents have active copies of the message that they will try to deliver. The probability that a communication failure occurs exactly at this step is directly proportional to the amount of filtering that the server performs on the message body, most often for anti-spam purposes. The limiting timeout is specified to be 10 minutes.

An agent present on the helpdesk server will be configured to sweep the alertbox email files as they arrive in the alertbox and send it to the alert display system and the enterprise data base for storage and later analysis.