**caCIS Reliability Improvement Approaches**

Following document provides various options which can be undertaken for improving the overall reliability and performance of caCIS software. These options are a mix of hardware, configuration and software changes which can be performed. Each of the options has associated pros and cons and also the time/effort taken. Based on these, the caCIS has created a suggest order in which these approaches can be applied to improve reliability gradually.

caCIS Team recommends the following four options

1. **Hardware Upgrades:** Upgrading the current hardware to provide more processing power and memory
2. **Deployment Architecture Changes:** Modifying the current singular deployment of caCIS into multiple nodes housing different components
3. **Mirth Connect Configuration Updates:** Modifying MirthConnect as well JVM (which houses Mirth Connect) settings for better reliability
4. **Code Changes:** Changing the code to primarily improve XML handling and processing as well as fixing memory leaks

The following subsection describes the changes recommended for each of these options in detail. caCIS team has also provided suggest order in which these options should be implemented to improve reliability without introducing too much risk to the project. These options will be reviewed with NCI/SAIC-F management and based on their recommendation applied to the caCIS project.

**Hardware Upgrades**

1. Following is the hardware setting for the virtual machine which is hosting the caCIS application

Dual Core 2+ GHz Processor

CentOS 5.7 x86\_64 GNU/Linux   
Java Version "1.6.0\_24"   
Java(TM) SE Runtime Environment (build 1.6.0\_24-b07)   
Java HotSpot(TM) 64-Bit Server VM (build 19.1-b02, mixed mode)   
CPU: Intel(R) Xeon(TM) CPU 2GHz x 1   
Mem: 6 GB   
Storage: 500 GB

1. All components (except for the Postgres database) of caCIS are running within this single VM. This can lead to shortage of computation power can cause memory shortages
2. caCIS team recommends upgrading the hardware to improve the reliability as well as overall processing capabilities of caCIS solution
3. caCIS team recommends the enhancing the current hardware settings to the following configuration

**Quad Core 2+ GHz Processor**

CentOS 5.7 x86\_64 GNU/Linux   
Java Version "1.6.0\_24"   
Java(TM) SE Runtime Environment (build 1.6.0\_24-b07)   
Java HotSpot(TM) 64-Bit Server VM (build 19.1-b02, mixed mode)   
CPU: Intel(R) Xeon(TM) CPU 2GHz x 1   
**Mem: 16 GB**Storage: 500 GB

1. caCIS team also recommends deploying and running caCIS on a dedicated single physical machine versus a virtual machine. This machine can run a **Quad Core Processor** which is powerful enough to handle all software that is part of caCIS. This way all the hardware capabilities of the machine are utilized just for caCIS components and not diverted towards running the virtual machine.

**Deployment Architecture Changes**

1. Currently all the components of caCIS solution (except for Postgres database) are deployed within a single virtual machine.
2. Following is the list of all the software components as well as the applications running on these software which form part of the caCIS solution:

Apache Tomcat 6.0.33   
Apache FtpServer 1.0.6   
Apache Maven 2.2.1   
Mirth Connect 2.1.1.5490   
PostgreSQL 8.4   
Virtuoso 6.1.3   
Subversion 1.6.11

1. As evident there are two primary software component which are processing and memory intensive
   1. Mirth Connect
   2. Virtuoso Database
2. caCIS team recommends splitting these two component into individual nodes as shown below

**Node A**

Apache Tomcat 6.0.33   
Apache FtpServer 1.0.6   
Apache Maven 2.2.1   
Mirth Connect 2.1.1.5490   
Subversion 1.6.11

**Node B**

Virtuoso 6.1.3

**Database Node**

PostgreSQL 8.4

1. This will allow both Mirth Connect and Virtuoso Database to utilize the processing power and memory as needed and without interfering with each other

**Mirth Connect Configuration Updates**

1. Mirth Connect provides JVM Options which can be used to provide various JVM options at the startup time.
2. These options can be used to control various features of JVM such as the Minimum and Maximum memory allocated to JVM, the PermGen memory allocated, Garbage collection settings etc.
3. caCIS team suggests setting these setting based on the upgraded hardware /deployment setup to provide maximum memory to Mirth Connect JVM
4. Apart from these Mirth Connect provides queuing within Channels. However currently it is provided only for certain destinations only (HTTP and LLP).
5. Even though LLP destination is used within caCIS, setting up queue for it wouldn’t help in improving the overall performance of caCIS as it will just queue messages which couldn’t be transmitted to the recipient.

**Code Changes**

1. caCIS deals with huge XML files both wile parsing as well as transformation. As a result of this it become very critical to improve XML handling and processing
2. One of the important improvements to use a single static JAXB Context. Creation of JAXB Context is a very resource intensive operation. Currently a JAXB Context is created for each and every request as shown below. This is undesirable as it adds unnecessary processing time and resource requirements.
3. caCIS team recommends creating the JAXB Context only once. This can be achieved by creating the JAXB Context as a static final variable as shown below

**package** gov.nih.nci.cacis.ip.mirthconnect;

:

**public** **class** CanonicalModelProcessorClient {

:

**public** CaCISResponse acceptCanonical(String wsdl,

String requestStr)

**throws** AcceptCanonicalFault, JAXBException, MalformedURLException {

:

**final** JAXBContext ctx = JAXBContext.*newInstance*(CaCISRequest.**class**);

:

}

:

}

1. Also another reduce unnecessary Marshalling / Unmarshalling XML data file which is present throughout the code (e.g. CannonicalModelProcessorClient)
2. Also apart from these there are mirth channel code changes which can reduce the overall memory footprint of caCIS solution. E.g.
   1. Clearing GlobalChannelMap variable after every request is completed
   2. Breaking the incoming caCIS Request into different components and passing/accessing only the components needed by a channel.

**Suggested Order of Changes**

caCIS team suggests the following order in which the above options should be implemented to improve caCIS’s reliability without introducing too much risk to the project.

1. Hardware Upgrades
2. Deployment Architecture Changes
3. Mirth Connect Configuration Updates
4. Code Changes