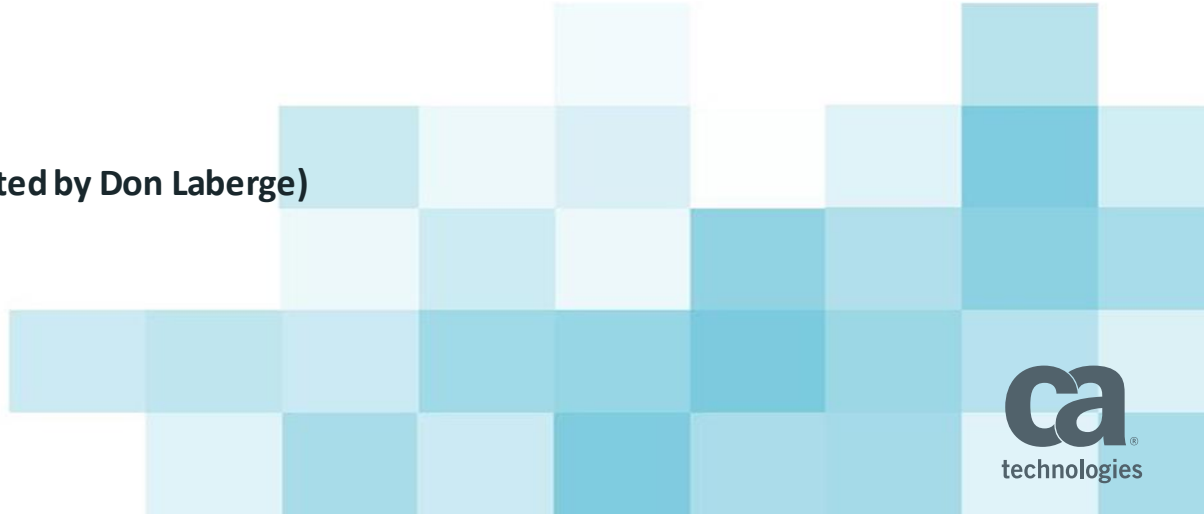
Spectrum
Fault Tolerant SANM

Karen Brooks
Principal Services Consultant

Created: 17 June 2016
Revised: 16 Aug 2016
Revised: 13 Oct 2016
Revised: 8 Mar 2017 (changes submitted by Don Laberge)



ca technologies

Agenda

1. SOLUTION OVERVIEW
2. HOW IT WORKS
3. PREREQUISITES
4. THE REST CALL
5. SCRIPT COMPONENTS
6. VALIDATION
7. QUESTIONS

Fault Tolerant SANM Overview

Fault Tolerant SANM Overview

- Supports distributed SpectroSERVER environment
- Notifiers from Primary and Secondary MLS can run simultaneously
- Supports Hot and Cold standby

How it Works

How It Works

- REST call queries the landscape to determine if the MLS is the primary or not.
- If the MLS is the primary, SANM will send through the primary MLS
- If the MLS is not the primary (failed over) SANM will send through the secondary MLS.
- Notifications from DSS are processed

Prerequisites

Prerequisites

- Spectrum 9.3+ (Web Services)
- OneClick servers *can* reside behind a load balancer. In this example, they are not behind a load balancer and the script queries the first successful OneClick connection and continues with the script.
- SANM Script data must reside on both the Primary and Secondary MLS servers.

The REST Call

The REST Call

Sample REST call results

We run a REST call to determine the value of the “isPrimary” value for the primary MLS. You will be prompted to enter valid OneClick credentials.

http://<oneclickhost>:<port>/spectrum/restful/landscapes

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<landscape-response total-landscapes="2"
xmlns="http://www.ca.com/spectrum/restful/schema/response">
        <landscape>
                <id>0x7d000000</id>
                <name>spectrum110</name>
                <isPrimary>true</isPrimary>
                <spectrumVersion>10.1.0.000</spectrumVersion>
        </landscape>
        <landscape>
            <id>0x7d200000</id>
            <name>spectrum111</name>
            <isPrimary>true</isPrimary>
            <spectrumVersion>10.1.0.000</spectrumVersion>
        </landscape>
</landscape-response>
```

The REST Call (cont'd)

'IsPrimary' tag

Here we determine the value of the 'isPrimary' tag for the MLS from the REST result. The value will be either true or false. **Note**: In testing, the 'name' tag will always reflect the name of the primary MLS even if 'isPrimary' is false.

The Primary MLS will evaluate 'isPrimary' for true

The Secondary MLS will evaluate 'isPrimary' for false.

```
<landscape>
        <id>0x7d000000</id>
        <name>spectrum110</name>
        <isPrimary>true</isPrimary>
        <spectrumVersion>10.1.0.000</spectrumVersion>
</landscape>
```

Script Components

Script Components

The SANM fault tolerant code is placed in the Set, Update and Clear scripts.

The beginning of the fault tolerant code is placed directly **above** the **echo_info()** line in the scripts.

The second half of the code is placed at the very end of the scripts **after** the following lines:

```
else
  echo_info
fi
```

Script Components (cont'd)

```
Beginning of SANM SCRIPT
...
...
First section of code
echo_info()
REST of SANM SCRIPT
...
...
else
  echo_info
fi
# END OF FAULT TOLERANT CONFIGURATION
exit 0
fi

done
```

Script Components (cont'd)

Pass the OneClick user password securely in the code

From the command line, enter the following command to encrypt the OneClick user's password:

```
openssl enc –base64 <<< mypassword
```

The encrypted password will appear:

```
bxlwYXNzd29yZAo=
```

In the $SPECROOT/Notifier directory, create a hidden file called .ftasv_notif (or something of your choice). Place the encrypted password inside the file. We hide the file just to make it a little more difficult for anyone poking around.

The following VARIABLES are used in the code to read and decrypt the password for the REST call.

```
NOTIF=`cat .ftasv_notif`
FTASV=`openssl enc –base64 –d –in <<< $NOTIF`
```

Script Components (cont'd)

The same block of code is designed to run on both primary and secondary servers

```
…
SANM Script Contents
…
NOTIF=`cat .ftasv_notif`
FTAUTH=`openssl enc –base64 –d –in <<< $NOTIF`

## create a lowercase / Hostname only to match what is returned by the rest call
TRIMHOST=`hostname | tr '[:upper:]' '[:lower:]' | awk 'BEGIN {FS="."} {print $1}'`

## Set the MLS LH
LH=0x7d000000

## Get the list of Servers - re-formatted one per line to a file
## to make it easy to parse out the entry we want
curl -s -o -X GET --basic -u myuser:$FTAUTH "http://<my-oneclick>:8080/spectrum/restful/landscapes" | sed -e
        s/\<landscape\>\<id\>/\\n\<landscape\>\<id\>/g > $$.temp
```

Script Components (cont'd)

```
## Get LH handle entry
RESULT=`grep ${LH} $$.temp`
rm -f $$.temp

## Does the Line contain true = 1 False = 0
COUNT=`grep -c "<isPrimary>true" <<< ${RESULT}`
MATCH=`grep –c ${TRIMHOST} <<< ${RESULT}`

## If Is Primary True (1) and Hostname = Primary MLS (1) = Primary Active Primary Forwards Alarm
## If IsPrimary False (0) and Hostname !- Primary MLS (0) Primary NOT active and Secondary Forwards Alarms

if [ ${COUNT} -eq ${MATCH} ]
then
    echo “This is not the Active SANM Server."

echo_info()

…
Remaining SANM Script Contents
…
```

Script Components (cont'd)

```
…
SANM Script Contents

FT_SANM Configuration

echo_info()

…
Remaining SANM Script Contents
…

else
  echo_info
fi

# END OF FT CONFIGURATION
   else
        echo "This is the Active SANM Server."
fi
```

Validation

Validation

When the Primary MLS is active:

The Primary MLS will display the following in its NOTIFIER.OUT:

This is the Active SANM Server.

When the Secondary MLS is active:

The Primary MLS AlarmNotifier is down, the Secondary MLS will display the following in its NOTIFIER.OUT:

This is the Not Active SANM Server.

Validation (cont'd)

Additional Validation steps

You may also wish to modify the $MAIL subject to include a server designator. This is useful when testing so that you can determine which server the messages are being sent from. You will also be able to determine any duplicates with this method by viewing the AlarmID value if it is being passed in the SANM data.

Primary MLS:

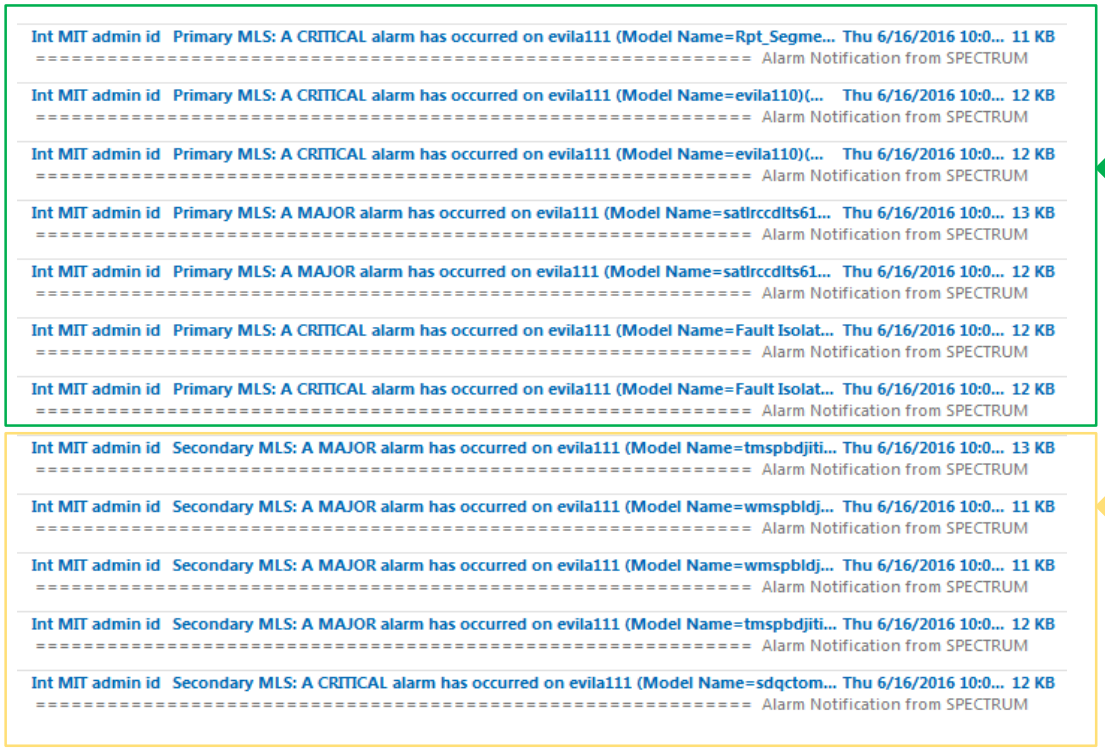
```
$MAIL -s "Primary MLS: A $SEV alarm has occurred on $SERVER (Model Name=$MNAME)(Model Type=$MTYPE)" $RCVRS < /tmp/set_alarm.$PID
    rm -f /tmp/set_alarm.$PID
```

Secondary MLS:

```
$MAIL -s "Secondary MLS: A $SEV alarm has occurred on $SERVER (Model Name=$MNAME)(Model Type=$MTYPE)" $RCVRS < /tmp/set_alarm.$PID
    rm -f /tmp/set_alarm.$PID
```

Validation (cont'd)

Real world example



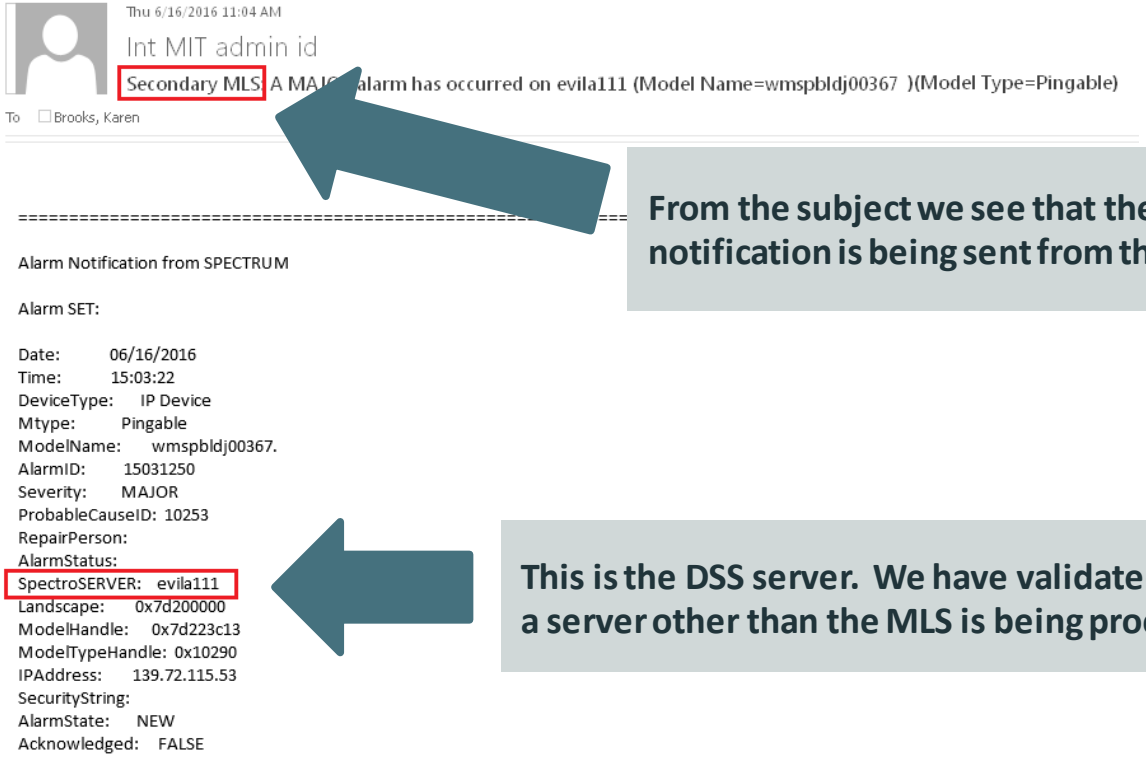
Running normally, notifications sent from Primary MLS

During failover, notifications sent from Secondary MLS

ca technologies

Validation (cont'd)

Real World Example - Notification received from DSS server:



From the subject we see that the notification is being sent from the Secondary MLS

This is the DSS server. We have validated that notifications a server other than the MLS is being processed.

ca technologies

Questions

Revisions

Created: 17 June 2016 - Karen Brooks:

- **Document creation**

Revised: 16 Aug 2016 - Karen Brooks:

- **'base64' command changed to 'openssl' so that is compatible across Linux and Windows platforms**

Revised: 13 Oct 2016 - Karen Brooks:

- **Updated code to have it continue after first successful connection to a OneClick server. If OC1 doesn't respond, it continues to OC2, makes a connection and resumes the script instead of also querying OC3.**
- **Primary MLS should not have any NOTIFIER.OUT output in a failover situation as the alarm service should be down.**

Revised 3/8/2017 – Don Laberge / Karen Brooks

- **Each MLS will no longer require a different file; one file can be used for both servers.**

Karen Brooks
Fault Tolerant SANM
Karen.Brooks@ca.com