HARMAN
A SAMSUNG COMPANY

Marimba by HARMAN

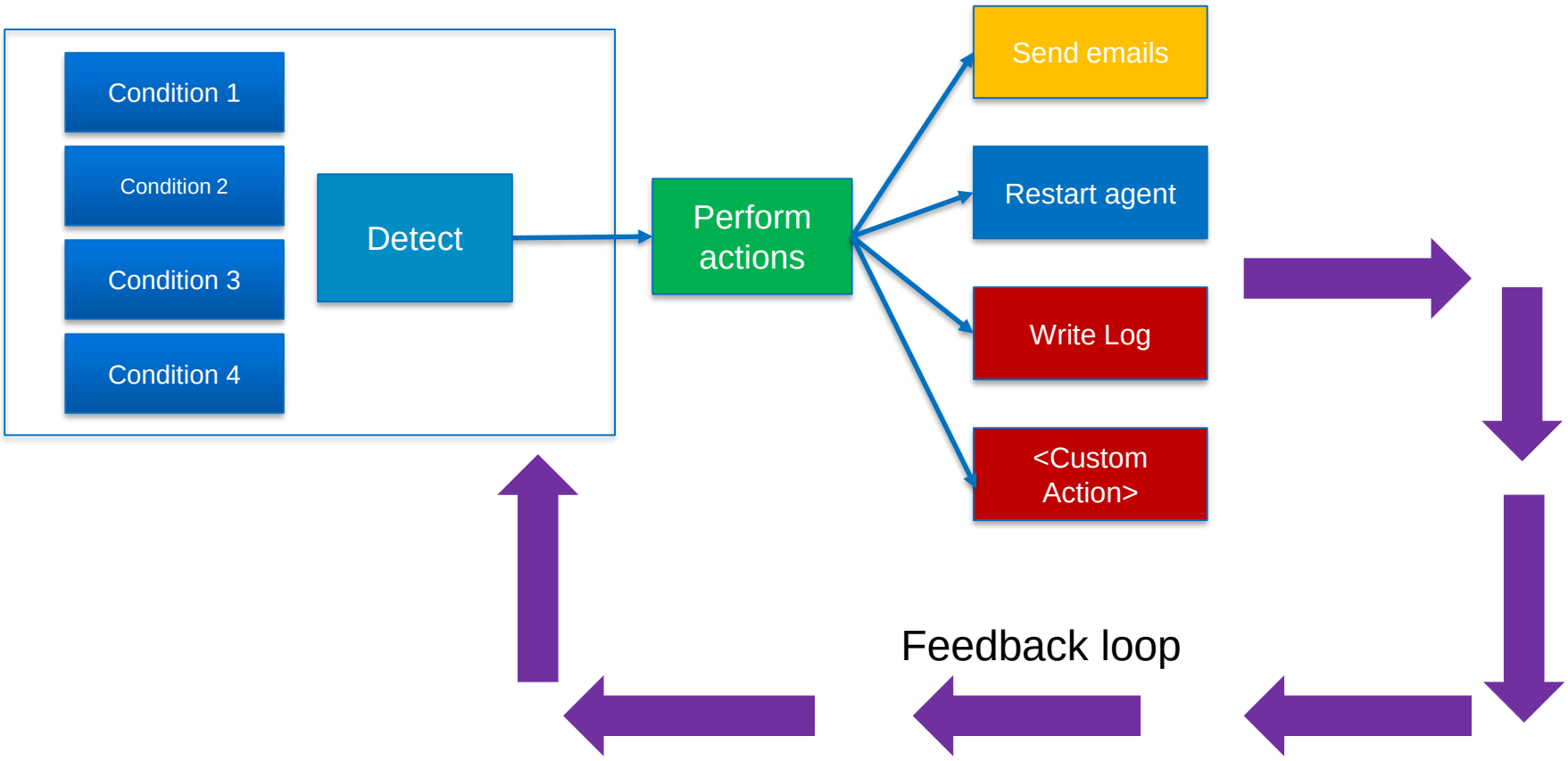
Modern, Scalable, and Secured Endpoint Management

Marimba User Group– June 2018

Nitish Shrivastava
Product Manager & Chief Architect

MaMoS – The Rationale

Intelligent Engine for Conditional-based execution *workflows*



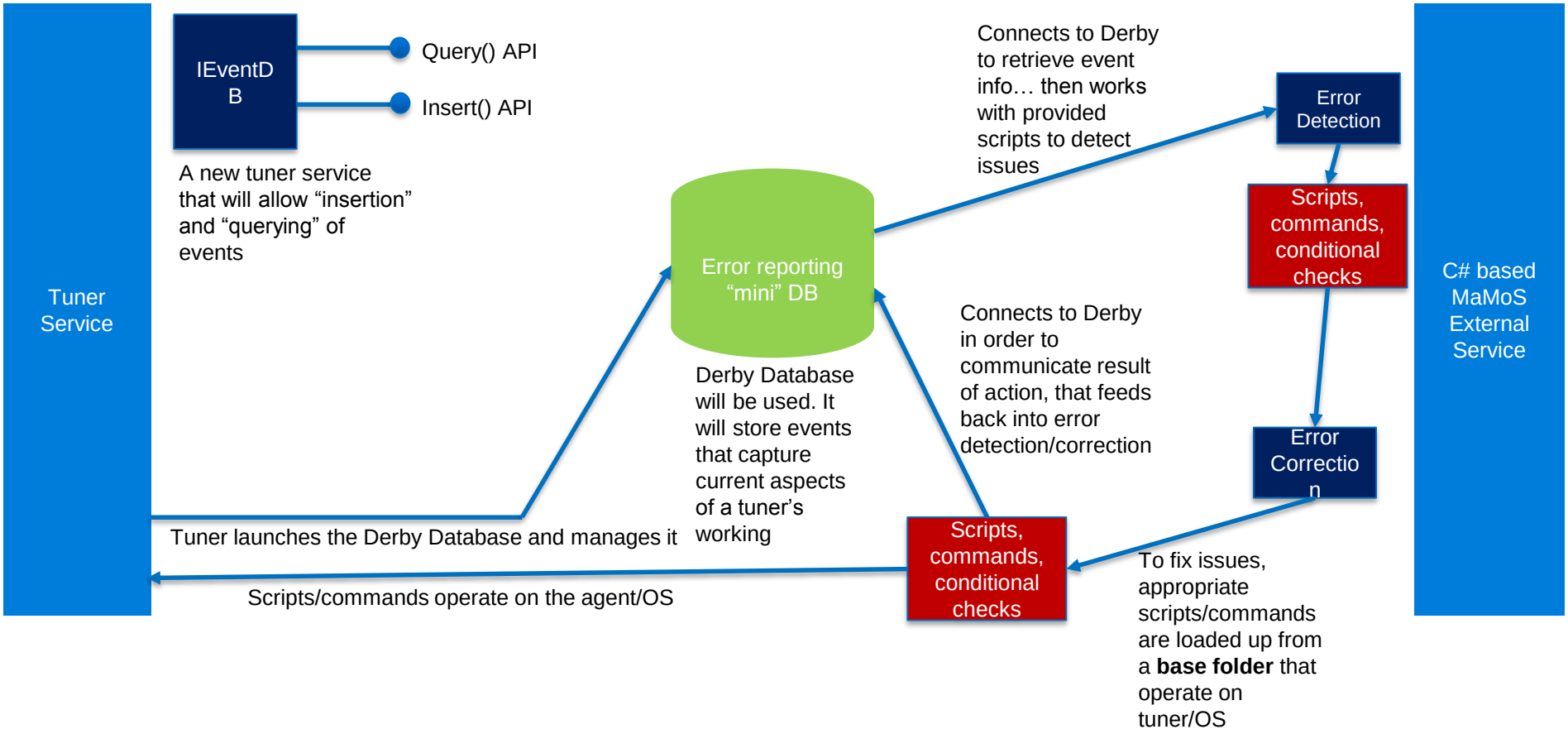
Designing for “Extensibility”

How do we ensure that MaMoS can fit all situations, and allow customization?

- *Make it a service that runs separately* from the agent
- Mode of operation
  - Once in a day (configurable via schedules)
  - Checks for “abnormal” conditions in agents
  - When detected, acts to remediate said problem based on pre-configured “instructions”
- Three main modules in the external service
  - Detection Module – Monitors the tuner/system based on available “condition” markers, and determines if the agent is “healthy” or not
  - Controller Module – Responsible for operational behavior (i.e. monitoring schedules, condition check triggers, logging, etc.), and provides mechanisms for providing feedback on operations
  - Action Module – Provides modules that processes configured instructions (written in a scripting language understood by the module), and triggers actions based on them. It preserves history of actions recorded, conditional execution, and many other such features
- These modules **work together to keep the agent/device running smoothly and in good health**

MaMoS Customization - Architecture

How the components work together

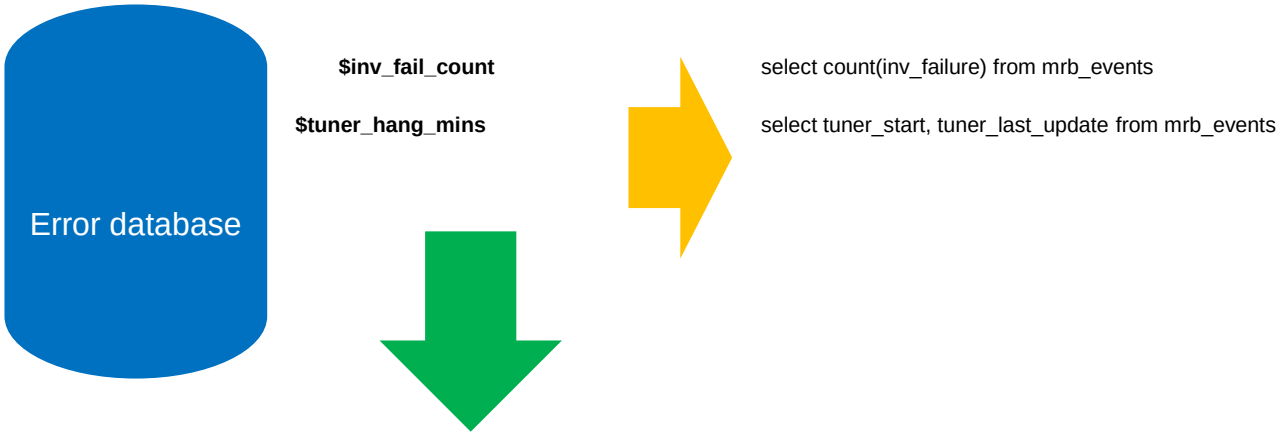


MaMoS – Configuring “Error Detection”

A Scripting Language for error *detection/correction* phases

Points

- You can configure "error detection" scripts:
  - A combination of macros/logic



$scan_fail >= 2 ? <problem> : <no-problem>

- This is delivered as an "error detection" script
- can incorporate multiple conditions:
- ($scan_fail >= 2 | $tuner_hang_mins >= 120 | $tuner_crash_count >= 2) ? <problem> : <no-problem>

MaMoS – Configuring “Error Correction”

Triggering/History of remediation logic scripts

Points

- configuring "error correction" scripts:
- macros for handling the "actions" to perform
- Roll those actions into the execution scripts for Tuner Stability

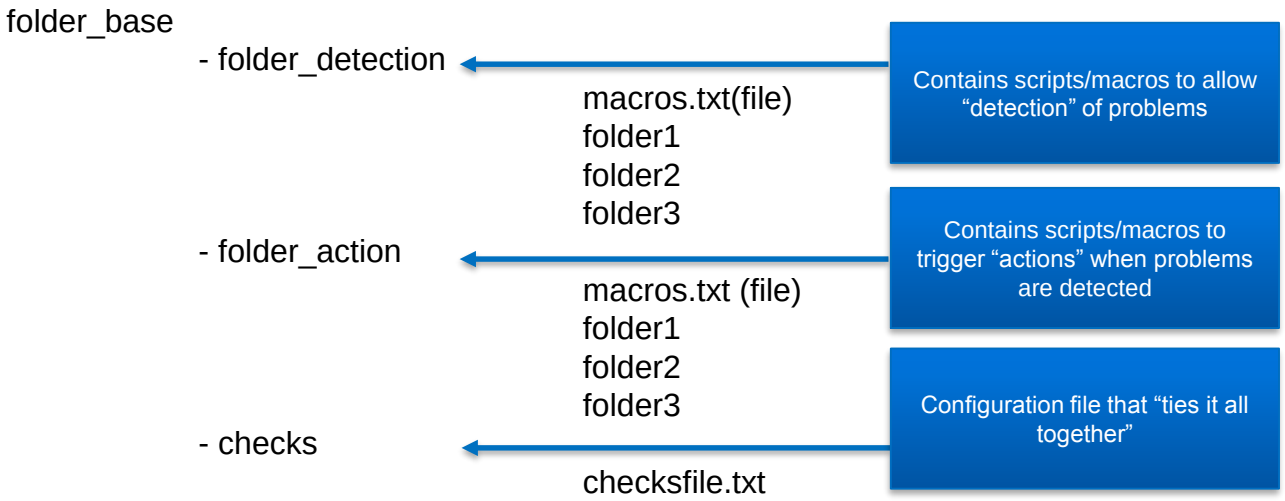
sc.exe start marimba	Msiexec -u <x> Rm –rf * Msiexec –I <x>	Tuner_ns <invsvc> -fullScan
$tuner_restart	$wipe_and_install	$force_inv_scan

($scan_fail >= 2 | $tuner_hang_mins >= 120 | $tuner_crash_count >= 2) ? $wipe_and_install : $force_inv_scan

Putting it all together

Folder structure for error correction/detection scripts

MaMoS functionality driven off of the contents of “base” folder:



Folder Structure Explained

Three folders contain the entire “logic”

- **“folder_detection” contents:**
- Macros.txt contents:
  <key>=<value> pairs, where
  key = macro name
  value = location of a file that contains combination of SQL statements and conditional logic for error detection

- Structure of the file:
  JSON format:
  [
  { "key1": "<sql/logic>"},
  { "key2": "<sql/logic"},
  ...
  { "keyn": "<sql/logic>"}
  { "result": "logic that combines previous key values... i.e. $key1 + $key2 > $key3"}
  ]
- The value of $result is passed into the macro "key". It can be used for subsequent actions

- **“folder_action” contents:**
- Macros.txt contents:
  <key>=<value> pairs, where
  key = macro name
  value = location of a file that either represents an executable, a batch file, or a tuner command

The actual files, BATs, executables, etc. can be dumped into the folders in this structure

- **“base” contents:**
- This folder will contain a file (checkfiles.txt) that encodes all the "checks" that the service has to do in order to check if the agent is healthy, and to take steps to do fixes.
- The checks themselves will be a series of lines representing the checks to perform.
- They will follow this structure:
  <condition>:<action>
  where:
  <condition> = combination of "detection" macros
  <action> = one "correction" macro

MaMoS Customizations - Feature

DEMO

Connect with us

Email:
info@marimbacloud.net

Address:
636 Ellis Street
Mountain View
CA 94043, USA

Phone:
+1 650 623 9400
+1 650 623 9401

Nitish Shrivastava
Product Management
Nitish.Shrivastava@harman.com

Arturo Samper
Sales
Arturo.Samper@harman.com

Ananda Waugh
Product Support
Ananda.Waugh@harman.com

Elaiyaraja T
Product Engineering
Elaiyaraja.T@harman.com

Corporate Website
www.services.harman.com

Product Website
www.marimba.com

Linked In
http://www.linkedin.com/groups/CMUG
com-Marimba-User-Group-1774296

User Group
http://cm-ug.com

Tuner Stability – Obtaining/configuring the feature

1. Getting the Tuner Stability feature

- Infrastructure Service – channel for delivery of the feature
    - Tuner upgrade of agents, subsequent configuration via tuner properties (more on this below)
- Works for both 32 bit and 64 bit Windows tuners
- Non-Windows support is imminent

2. Configuring Tuner Stability

- Driven by tuner properties
    - marimba.tuner.mamos.enabled=true
        - What does this do: Enables the main feature
        - Default value: false
    - marimba.tuner.mamos.eventdb.port=[integer]
        - What does this do: Specifies the port on which the event DB service “listens” for requests to update/query event information about the installed Marimba agent
        - Default value: none (so must be set explicitly!)
    - marimba.tuner.mamos.base=[path to a folder]
        - What does this do: Must specify the “base” location in which scripts/macros for operation are present. This includes BOTH the “out of the box” scripts that the feature will ship with, and any customizations that customers intend to do. So in other words, any custom scripts, BAT files, etc. must go into this folder in order for the feature to “recognize” and apply them.

Tuner Stability – Obtaining/configuring the feature

2. Configuring Tuner Stability (continued)

- More tuner properties
  - marimba.tuner.mamos.checkinterval=[long value]
    - What does this do: This denotes the interval in which the mamos external service will “apply” the detection/correction scripts in the base folder
    - Default value: 3600000 (in milliseconds). Essentially defaults to an “hourly” check
  - marimba.tuner.mamos.idlookup=[path to a lookup file]
    - What does this do:
      - The event DB is meant to be a repository for all “events” that have taken place in the tuner.
      - By default, it will capture **everything** (i.e. every log ever generated, be it tuner start/stop, or inventory scan related messages, policy application, etc.). This can be a lot
      - The aim of this property is to “filter” out what gets captured. You can have a file containing key=value pairs, where the key must be a log ID that’s “to be captured”, and the value must be “true”
      - Example of entries:
        - 1001=true (i.e. log messages with log ID 1001 will be captured)
      - Note: If you have this property configured, then ONLY the log IDs in the lookup file will be captured. So if you use this and you have queries that depend on the existence of records related to certain log IDs, then those log IDs MUST be present in the lookup file pointed to by this property.
    - Default value: none (in other words, “no restrictions, capture everything”)

Tuner Stability – Obtaining/configuring the feature

2. Configuring Tuner Stability (continued)

- More tuner properties
  - marimba.tuner.mamos.extdb=[string]
    - What does this do: This denotes the folder *on the filesystem* in which the event DB service is installed. Note that this *copies* out the contents of extdb folder into specified location.
    - Default value: None (so it is a mandatory setting, i.e. “C:\extdb”)
  - marimba.tuner.mamos.extsvc=[string]
    - What does this do: This denotes the folder *on the filesystem* in which the external monitoring service is installed. Note that this *copies* out the contents of extsvc folder into specified location.
    - Default value: None (so it is a mandatory setting, i.e. “C:\extsvc”)

Tuner Stability – Obtaining/configuring the feature

3. Tuner Stability operation

- Sequence of events
  - Until the properties are set, all that happens is that the binaries are “staged”
    - In the lib folder of the tuner, you will see three folders, “extsvc”, “extmon” and “base”. These correspond to the binaries for the Mamos external service, the event DB, and the base folder for scripts respectively
  - When properties are enabled
    - On tuner restart, these binaries are “installed”
      - extmon is copied out (along with a copy of the JRE for Derby) into configured folder, and the event DB is installed as a service
      - extdb is copied out (essentially a single executable with some DLLs), and the external service is installed
      - base folder is NOT copied out (*this is to ensure we don’t overwrite a customer configured script*). Customers can configure their base folder in the location pointed to by the marimba.tuner.mamos.base property (a “reference” base folder will be provided to customers as a zip. **When this feature is released, we will deliver this via an updatable channel**).
    - Components “switch on” and start working
      - Tuner starts pushing events in real-time to the event DB
      - The external service periodically triggers checks (based on configured interval)
        - Looks at base file, loads master script
        - Accordingly locates configured error detection scripts, runs them
          - Checks if “threshold” for scripts have been reached
        - If a “threshold” is breached
          - Kick off the configured error correction script/executable/BAT
  - Where’s the tuner in all this?
    - Firing events to the event DB
    - Managing update/install of the Tuner Stability services
    - Minding its own business (policy updates, inventory scans, patch installs, etc.)

Tuner Stability – Obtaining/configuring the feature

3. Tuner Stability – “Out of the box” error detection/correction scripts

- Tuner will ship with out of the box scripts for some error detection/correction scenarios
  - Tuner “hung” (encompassing tuner logging stopped, channels not updating, etc.)
    - Correction scripts available: forceful tuner restart, force trigger of channels, etc.
  - CPU usage continuous spike
    - Correction scripts available: forceful tuner kill/restart
  - OOM issues
    - Correction scripts available: forceful tuner kill/restart
  - Patch corruptions/issues
    - Correction scripts available: Patch info verify/update, force Patch Service start
- But it’s all configurable
  - Modify the existing scripts
  - Roll your own scripts
    - Add executables, BAT files
    - Add more events via custom channels