

1. Overview

Sapphire core work have two big categories.

1. Multiple DM chaining
2. Object life cycle management, including object sharing and object placement

Schedule

https://docs.google.com/spreadsheets/d/12xdBfFyTDCIkWn-jjjHTOM_8FSursaGfSpu06_WtxMY/edit#gid=459705536

Task in Phase 1

1. Object retrieval by URI.
 - a. See [Life Cycle Management] - <Object by its name> section for details.
 - b. Support URI format to retrieve the objects created before.
2. Object health check by Kernel.
 - a. See [Object health check and management] topic for details.
 - b. Kernel Server should check the status of object and report to OMS.
 - c. Update example app to use this feature (?).
3. Investigate DMs' implementation on object life-cycle management - particularly group policy. Update as needed for missing part. Implement to use newly added OMS API if needed.
 - a. See [Object health check and management] topic for details
 - b. Go over each category to see if DMs already have implemented it. Update the document with confirmation detail.
 - c. Add life-cycle management logic in DMs if they are missing and needed.
 - d. Implement a simple life-cycle management DM with focus on group policy to be a good general solution for most of DMs to reuse. (TBD) Consider the use for multi-DMs chaining.
 - e. If there are too many missing parts in DMs, would it be better to implement on Sapphire core?
4. Locate Group Policy in OMS.
 - a. See [Placement of Group Policy] topic for details.
 - b. Add a choice to Group policy to run on OMS instead of Kernel Server. Support to run at a separate Kernel Server as well?

Pending Questions

1. (5/23/2018) Should we remove the concept of app entry point? What is the negative impact of removing app entry point? We can give every Sapphire object a dns name and allow developers to retrieve sapphire object from OMS with its dns name.
 - a. (5/24/2018) OMS requires to specify app entry point. Can we avoid specifying the app entry point when launching OMS?
2. Should DM be in charge of the monitoring of Sapphire object health and the creation/deletion of Sapphire objects, or should kernel server do it? Or some kind of responsibility separation between DM and Kernel Server?
3. (5/23/2018) Conflicting roles between DM and central life cycle management by KS/OMS.
 - a. Should we expect garbage collection from DMs? If so, we may have to reimplement all of the existing DMs. We can revisit this item once we have a central garbage collection system (or distributed by KS).

Object Life Cycle Management

Priority: 1

Owner: TBD

Estimate: 6 weeks (1-2 devs)

Requirements

Life Cycle Management

- P1: Manual Object Creation and Removal
 - Applications can create and remove Sapphire object via explicit calls to OMS/Kernel
 - (5/25/2018) The following existing PR (ScaleUpDownDM) already provides most of the removal implementation:
<https://github.com/Huawei-PaaS/DCAP-Sapphire/pull/96/files>
 - (5/25/2018) Make sure SO no longer exists after removal. Java RMI garbage collection:
<https://docs.oracle.com/javase/8/docs/platform/rmi/spec/rmi-arch4.html>
 - Be able to create Sapphire object without using entry point
 - Today OMS is launched with one App Entry Point, and all Sapphire objects are created by App Entry Point. Why is that? What is the purpose of creating App Entry Point? As a Sapphire user, I just need a special *new* operator with which I can create any Sapphire object at any time, and

later on retrieve it with its name. Is there any negative impact to get rid of App Entry Point?

- P1: Object retrieval by its name
 - Being able to retrieve the existing live object (crashed object should follow the protocol specified in monitoring and recovery in other sections).
 - Typical use cases include:
 - After device reboot, the application may want to fetch the Sapphire object it created before reboot.
 - The application on one device wants to use the Sapphire object created by the application running on a different device.
 - Support URI like format to retrieve the object.
 - https://en.wikipedia.org/wiki/Uniform_Resource_Identifier
 - URI should
 - Be a globally unique identifier.
 - Has friendly name.
- P3: Automatic Object life cycle management
 - DCAP Creates and garbage collects Sapphire objects automatically based on specifications from applications.
 - (5/24/2018) Note: In general, application is responsible for removing SO. Garbage collection should be a fall back plan.
 - Example scenarios
 - (Optional or unnecessary, 5/24/2018) DM (or application?) may say Sapphire object A should be created on demand and be recycled after being idle for 5 mins. KernelServer may need to support this by providing an API call from Group Policy at OMS. This may be total responsibility of DMs. In such a case, we may want to provide pluggable DM that will support this in addition to the existing DM. Further investigation on whether non-DM side operation will be necessary is required.
 - DM (or application?) may say Sapphire object B should be long running. In either case, DCAP will make sure to create and garbage collect Sapphire object properly.
 - (+5/18/2018, optional) In case DM does not clean up the objects properly, KernelServer itself may need to be able to collect the garbage objects.
 - Creation
 - Instantiate Sapphire object initially
 - Most are already available so no need for additional work.
 - Anything else?
 - Failure Recovery
 - Sapphire object failure recovery is one common use case. For *long-running managed* Sapphire objects, DCAP should

be able to detect object failure, and recreate object promptly.

- (+5/18/2018) Server failure vs object failure
 - Server failure: who will restart kernel server? It is out of scope of this project. User may choose to use their own solutions.
 - Object failure: Depending on specification from application, monitoring alert should be able to trigger object recovery. (+5/21/2018) Object crash while KernelServer is running fine is rare. (5/25/2018) We plan to separate Kernel server and SO as separate processes after . Therefore, this is a likely scenario.
- (+5/18/2018) Recovery methods (two options):
 - (Probably not needed, 5/24/2018) Restart the object (simple) - This should be decided by DM.
 - Retrieve the object state (latest state stored) from storage (internal or external).
- Removal
 - Sapphire object garbage collection is another common use case. DCAP should be able to garbage collect *idle* Sapphire objects based on application's specification.

Health check and management (P1)

- Monitor the health of Sapphire objects by checking the signal sent by Kernel Server and reports/logs the information.
 - P1 (5/21/2018) OMS or a separate monitoring process can be used. (5/25/2018) For the first iteration, we use OMS with a separate thread. OMS gathers the information sent from each Kernel Server periodically.
 - P5 (5/24/2018) Further discussion needed for "Object status".
 - Examples:
 - ~~Not found:~~
 - ~~No longer available: crashed or had an issue but not recreated:~~
 - ~~Started: object was instantiated and has been healthy since the inception (never crashed):~~
 - ~~Recreated: object has a history of crash and it was recreated (instantiated again):~~
 - ~~Recovered: object has a history of crash but recovered with previous state:~~

- P3 (5/24/2018) ~~Client can request OMS (or central monitoring process) to send the monitoring result by calling its API.~~
 - P4 (5/24/2018) ~~May not be need: OMS (or central monitoring process) can propagate the monitoring messages to subscribed channels.~~
- Define a health check API in Sapphire object interface.
- (5/24/2018) Need a Kernel Server to ping health check API periodically to detect object failures. This component can be Kernel Server which checks the local So's, and DM.GroupPolicy (running inside OMS) which checks Kernel Server (and thereby learn health of its SO's).
 - Scenario
 - OMS crashes.
 - It may not need health check mechanism because restart of OMS is out of scope and restart of OMS should take care of it.
 - Instead, it should to be able to read the server list from storage to get the latest version of healthy servers.
 - (5/25/2018) Active monitoring of kernel server by OMS is debatable for the need. Therefore, it is pushed out and should be revisited later after other components are done.
 - Kernel Server crashes and so do objects associated with it.
 - (5/24/2018) Restarting Kernel Server is out of scope. This scenario is what to do when Kernel Server was restarted after crash. Deployer can use initd, systemd, Kubernetes etc.
 - (5/24/2018) Depending on the scenario and configuration of each object, do either of the following. These should be done by Group Policy; therefore, OMS should provide API only to support this (5/25/2018).
 - Do not start the object until it is invoked by the request from app (P1). OMS.DM.Group policy decides and it services onRefRequest().
 - Restart the object automatically and recover the object Id (possibly by asking OMS (P2) or reading from file system/DB(P3)). Set the status as crashed until it is restarted. OMS.DM.GroupPolicy can do this and it services onMembershipChange().
 - Save the object snapshot periodically, and recover from the snapshot if that was the desired behavior. If unrecoverable, either restart the object of wait until it is invoked by the app. (P4 or never)
 - Object crashes but Kernel server is behaving fine.
 - ??? (5/25/2018) Kernel Server needs to restart the object, possibly by using a signal handler as SO will be the parent

process of KernelServer, so it will be notified when any of its child processes crashes/corrupted. This decision should be based on Kernel Server's monitoring health of SOs such as deadlock, bad health status.

- OMS

- Note it is NOT object health check but Kernel server health check.
- We may not need the health check by OMS if user controls the server management overall. However, OMS should be able to ~~remove~~ mark the server as unreachable from its list of server table upon request (by whom? Script to call OMS to support integration with container/kubernetes?).
- OMS checks the health of Kernel Server periodically.
 - Kernel Server sends heartbeat to OMS. ~~Remove~~ Mark as unreachable if unresponsive after a few retries.
 - OMS pings each Kernel Server in the list. ~~Remove~~ Mark as unreachable if unresponsive after a few retries.

- (5/25/2018) Kernel Server monitors its object

- This topic is covered in detail at above scenario section. Please refer to the section.
- P2: Notification (?)
 - Need a mechanism to notify object life cycle events. Check that this is doable from DMs first by using onCreate(), onMembershipChange() etc.
 - Events should be sent out whenever an object is created, removed, migrated, etc. This is responsibility of DMs. Hence, check the existing DMs and confirm they handle this.

Object Location Management (P3)

- This should be handled by DMs (OMS.DM.GroupPolicy). Check whether current DMs are doing this properly and we can ignore this.
- Overlap with DM's responsibility:
 - 1) annotation in DM to mark the location or
 - 2) linked DM chain that will include a DM that supports location management.
- (tentative - verify we already have these DMs): Applications should be able to specify location preferences of Sapphire objects, e.g. keep in Cloud, keep on devices, stay in one AZ, etc. This might better be implemented on DM.
- (tentative): DCAP decides where to put the Sapphire object upon object instantiation or object resurrection, but it must respect the rules specified by applications.
 - Annotation on Sapphire object? Investigate whether this is needed once DM support annotation for identifying the location.
 - DMs, e.g. Code Offloading DM, may migrate Sapphire object, but the migration must respect the rules specified by applications.

Object Sharing (P3)

- P3: Permission control: It might be better to implement on DM side. Investigate.

- Others are related to object retrieval and recovery in above requirement.
- (5/24/2018) Use URI to access globally shared object.

(5/25/2018) Placement of Group Policy (P2)

- Group policy should be able to co-locate with OMS (preferred) or locate in a different kernel server.

Questions

- Q: do we want to separate Group policy from Server policy (in a different location)? E.g., maybe group policy should be in a separate server such as OMS? Group policy should be fault tolerant.
- Etcd for fault tolerance?
 - RAFT inside OMS? Preferred once RAFT becomes stable.
 - External Etcd cluster?

(5/24/2018) Enhance Stability of OMS and KernelServer (P3)

- Use of persistent storage
 - OMS
 - Use local/cloud file system or local/cloud database to store object mapping table and status?
 - Put on local disk with replication using RAFT.
 - Kernel Server (may not be needed).
 - Use local/cloud file system to save the snapshot of object state.?
 - May not be necessary as SO will be gone when Kernel Server crashes. The decision for this this should be at OMS.Dm.GroupPolicy.

Other Topics

(5/25/2018) Integration with Kubernetes

- Monitor/Recover OMS & Kernel Server pods/container using Kubernetes.
 - Investigate example scenarios/demos.
 - Provide script to demonstrate the scenario.
 - Implement any missing APIs in Sapphire core.
 - Provide a document for easier on-boarding.
- Integrate monitoring/life cycle management at Sapphire with Kubernetes.
 - Send health check signal to Kubernetes controller.
 - Provide decision criteria and execution commands for each signal.
 - Implement any missing APIs in Sapphire core.

Tasks (Life cycle management)

Days	Task
	Explicit Removal
1	1) OMS and KernelServer API to remove requested Sapphire objects (per object & per app) based on provided URI or UUID.
(1)	2) Optional: save the object in storage (internal/external). We already have DM for this. Consider plugging in to Tapir/Diamond.
	Explicit Creation
1	1) Create Sapphire object directly on Kernel server without using app entry point.
1	2) Support lazy registration to OMS (e.g., retry to register object).
	Object Retrieval
2	1) Previously created object can be retrieved by the request from application.
3	2) (optional) Object can be saved in storage (internal/external) for later retrieval or crash proof.
	Object Information Retrieval
2	Object information can be queried to OMS directly for: • Object ID referenced currently • All objects for application or group or region • E.g., In which region is the object located? Query the object ID of the currently used Sapphire object. Obtain a list of all object IDs that are created so far for the same class - this is needed to check to see if there are unused objects (should have been garbage collected or explicitly removed). Following information can be queried: • Timestamp • Status (Recreated, Unavailable, Crashed etc). (+5/23/2018) This task has some duplicates with object sharing tasks. Please refer to the object sharing so the retrieval implementation can allow the object sharing logic.
	Object Monitoring
1	1) (5/25/2018) Kernel Server sends monitoring signal to OMS. The result can be accessible by client. (OMS or central monitoring process) can check the health of Kernel Servers manually by client request. E.g., client application call OMS API to get monitoring result.

1	2) (OMS or central monitoring process) can check the health of objects manually in Kernel Server by client request. E.g., client application call OMS API to get monitoring result.
3	3) (5/25/2018) (Decoupled monitoring process in OMS) can check the health of all kernel servers or kernel servers with objects automatically (e.g., periodically triggered) based on the signals sent from Kernel Servers.. Status of objects (or Kernel Servers?): • Not found. • Starting. • Running. • Shutting Down. • No longer available: crashed or had an issue but not recreated. • Started: object was instantiated and has been healthy since the inception (never crashed). • Recreated: object has a history of crash and it was recreated (instantiated again). • Recovered: object has a history of crash but recovered with previous state.
3	P5: 4) Broadcast the status to a message queue where any clients or kernel servers can subscribe.
	P5: 4) (KernelServer? : optional) can check the health of objects manually.
	P3: 5) (KernelServer? : optional) can check the health of objects automatically (e.g., periodically triggered).
	Object Health Management (Recovery)
3	1) Subscribe the monitoring status of each object and recreates it if specified in the configuration. Report back the status change.
3	P5: 2) Object recovery: • Mark the object as recovered and log the timestamp of recovery. • Application should be able to use these properties to make a decision. • Option to save the object status in the local/external storage.
	Object Notification Subscription
3	1) P5: Application can subscribe to a channel where object status is informed (e.g., dead, recreated, timestamp).

2	P4: Investigate whether the following is available: 2) ??? Kernels can subscribe to notification channel which can pass the information to DMs.
---	---

Tasks (Garbage collection: P3): Note there is significant overlap with monitoring and health management. This is pushed until both tasks are complete.

Days	Task
3	Decision at OMS (P4) - further discussion needed whether this is needed. Monitoring will cover health check part. OMS checks client application whether it is reachable for Sapphire object that allows garbage collection on client network connection failure (some objects may need to live even when no clients are alive for later use with preserved state). Client reachability is checked multiple times with exponential back-off. For shared object, it checks the reachability to all the clients that are using the object. If confirmed to remove the object, it executes removal plan. • OMS may have a table to denote last accessed timestamp to clean up the expired object. Application client can specify this (OMS may put default expiration time otherwise).
2	Decision at Client (P3) Clients send signal to OMS or Kernel Server to remove the object directly once the reference to the object is no longer available (e.g., out of code-path - JVM determines it cannot find the object).
3	Execution at OMS (P4) OMS removes the objects in all of the servers including replicas. If removal fails, roll back (recreate the object) or don't do anything except logging and issuing critical error to notification channel (optional).
2	Execution at Kernel (P3) (5/23/2018) Kernel removes the object based on information obtained from OMS by contacting it (and all of the replicas if it cannot be done in OMS due to DM specific operation). For example, if replicas were created by DM (e.g., Master-Slave DM or LoadBalanced DM), Kernel server may not know the location of objects. Therefore, Kernel Server should contact OMS to locate all of the objects. If removal fails, roll back (recreate the object) or don't do anything except logging and issuing critical error to notification channel (optional).

Tasks (Object sharing: may be duplicate with other tasks except permission control)

Days	Task
------	------

3	Permission Control (P3) Each object can be shared based on permission specified by the application that created an object (by requesting to OMS): • Per application • Per group membership • All application
	Single Object Sharing Implement APIs and tables to retrieve the shared object per app and per object for single object. • Object IDs to application name table. • Object IDs to shared group name table (membership group name). • (optional) Object IDs to shared set of clients table (list of clients). • (optional) Or maybe group names determined by the postfix of method name (e.g., _UsEast to allow access from US East region by region or client IP range). Upon request by application to OMS, OMS checks the internal object table, and returns the reference to previously created object if shared property is mark.
	Failure Handle • Use object health management • Configurable decision for failure handle ◦ Recreated all the time. ◦ Inform OMS but wait until client makes specific query. ◦ State preserve in local storage for recovery (use object health management)
	Multiple Replicas Implement APIs and tables to retrieve the shared object per app and per object for multiple replicas of an object. • Investigate duplication with DM side of replicas: is object sharing from non DM side even needed? • Group of objects can be created over multiple servers or on a single server. • Assign the object to each application in a random (or round-robin) fashion first (as the request comes from application). More options can be added later.

	Shared Group • Can mark/unmark the object as shareable upon creation or after creation (basically anytime). • Configuration will have the options to be shared ○ To applications ■ all application ■ set of applications ■ same application ○ To members ■ Region based ■ Private group ■ Explicitly set clients (e.g., set of IP addresses or range of IP addresses)
	Object Removal Implement manual removal process and let garbage collection use it for removal of shared objects. • OMS checks all the groups that have mark in the table as currently using it. • (optional) ○ Optional flag to preserve state in internal or external storage (check with object removal section in life cycle management). ○ Optional flag to allow removal based on majority voting. ○ Optional flag to allow removal from a main application (or DM?) ○ Where to locate the flag? OMS table? Sapphire object?
	Access restriction • Object sharing based on encrypted key?

Tasks (Location Management)

Days	Task
	Implement OMS APIs to locate the objects to different servers (e.g., servers in different region). Reference explicitMigrationDM and codeOffloadingDM (not implemented yet).
	Implement OMS trigger APIs to move the objects (user configurable) • Network latency • Memory • CPU

Task : Creation of sapphire group policy instance on OMS

Introduction:

At present, group policy instance of DM for the particular sapphire object is created on one of the kernel servers similar to the server policy instance. But, Kernel server are not fault tolerant. OMS is supposed to be fault tolerant. Hence group policy instances can be created on OMS so that group policies are ensured to be more reliable.

Process Flow:

1. Define a createGroupPolicy method in OMSServer interface.

```
SapphireGroupPolicy createGroupPolicy(Class<?> policyClass) throws RemoteException, KernelObjectNotCreatedException, ClassNotFoundException
```

2. Make getPolicyStub and initializeGroupPolicy methods defined in sapphire.java as public such that they can be accessed from createGroupPolicy method.
3. Provide an implementation of createGroupPolicy in OMSServerImpl class as shown below:

```
@Override
public SapphireGroupPolicy createGroupPolicy(Class<?> policyClass) throws RemoteException,
KernelObjectNotCreatedException, ClassNotFoundException {
    SapphireGroupPolicy groupStub = (SapphireGroupPolicy) getPolicyStub(policyClass);
    try {
        initializeGroupPolicy(groupStub);
    } catch (KernelObjectNotFoundException e) {
        logger.warning( msg: "Failed to find the group kernel object created just before it. Exception info: " + e.toString());
    }
    return groupStub;
}
```

4. Create an instance of kernelServerImpl in OMS and export it to registry when OMS is being initialized in main method.

```
/* Export kernel server service */
KernelServer localKernelServer = new KernelServerImpl(new InetSocketAddress(args[0], Integer.parseInt(args[1])), oms);
KernelServer localKernelServerStub = (KernelServer) UnicastRemoteObject.exportObject(localKernelServer, port: 0);
registry.rebind( name: "SapphireKernelServer", localKernelServerStub);
```

5. When sapphire object is being created with new_ operator in kernel server, invoke createGroupPolicy method on OMS(RPC call) so that group object(including its OID allocation) happens on kernel server instance within OMS and returns a groupStub instance to the kernel server creating sapphire object. This returned stub can be injected to sapphire server stub and client as we do currently. when client or server policy tries to call group policy methods such as onCreate/addServer/getServers/onFailure/onRefRequest, RPCs are routed to kernel server within OMS process.

Old Code in new_ method:

```
/* Create and the Kernel Object for the Group Policy and get the Group Policy Stub */
SapphireGroupPolicy groupPolicyStub = (SapphireGroupPolicy) getPolicyStub(sapphireGroupPolicyClass);

/* Initialize the group policy and return a local pointer to the object itself */
SapphireGroupPolicy groupPolicy = initializeGroupPolicy(groupPolicyStub);
```

```
groupPolicy.onCreate(serverPolicyStub);
```

New Code in new_ method:

```
/* Create the Kernel Object for the Group Policy and get the Group Policy Stub from OMS */  
SapphireGroupPolicy groupPolicyStub = GlobalKernelReferences.nodeServer.oms.createGroupPolicy(sapphireGroupPolicyClass);  
groupPolicyStub.onCreate(serverPolicyStub);
```

Note: onCreate is an RPC call now.

6. SapphireGroupPolicyUpCalls interface methods are to be modified to throw Exception. Modification of places using these methods are not shown here.

```
interface SapphireGroupPolicyUpCalls extends Serializable {  
    void onCreate(SapphireServerPolicy server) throws Exception;  
    void addServer(SapphireServerPolicy server) throws Exception;  
    void removeServer(SapphireServerPolicy server) throws Exception;  
    ArrayList<SapphireServerPolicy> getServers() throws Exception;  
    void onFailure(SapphireServerPolicy server) throws Exception;  
    SapphireServerPolicy onRefRequest() throws Exception;  
}
```

Multiple DM chaining

Priority: 0

Owner: Sungwok Moon

Estimate: 4 weeks (1-2 devs)

Requirements

Multiple DMs can be used per Sapphire object.

- Client/Server/Group actions in each DM should be executed as in a single DM case.
- Client DM may call the server DM directly or call RPC in its client parent class.
- Design principle should be intuitive and straight forward rather than many work around (or difficult concept).
- Identify the ways to prevent use of conflicting DMs.

TODO

Break down the tasks into smaller pieces with more detail as we learn more.

Tasks

Days	Task
3	Come up with a design that enables multiple DMs.

5	Implement happy path for direct RPC call from client DM to server (e.g., load balanced DM).
(3)	Implement to support RPC call to parent client classes before calling the DM in server (e.g., DHT policy DM).
2	Handle failure scenarios that might only happen in chained DM scenarios (may require update on OMS/Kernel). - Prevent the use of conflicting DM?
0.5	Test simple two DM scenario with one of the example apps (e.g., HanksToDo).
1	Test multiple DMs with Core DMs such as Consensus/Distributed Transaction DM.
0.5	Test the client applications both on mobile and desktop.
0.5	Test when Sapphire core is running on mobile and desktop.
0.5	Identify the missing scenarios and use cases.
3	Complete the missing/additional parts.
15+(3)	TOTAL

TBD

- Client to server call sequence (DM1.client->DM1.server or DM1.client->DM2.client).
- What to do in case of failure?
 - Should there be a separate operation for DM chain exception scenario?

Inheritance model (optional)

- Identify when this model will be useful.

MISC

- Preliminary investigation revealed most of the changes may happen in Sapphire run time.
- Happy path sequence
 - Client: beforeRPC() on DM1
 - Client: beforeRPC() on DM2
 - Server: beforeRPC() on DM2
 - Server: beforeRPC() on DM1
 - Server: Invoke app object, return the result.
 - Server: afterRPC() on DM1
 - Server: afterRPC() on DM2
 - Client: afterRPC() on DM2
 - Client: afterRPC() on DM1

Meetings

2018-5-16 (Tasks for each goal for initial design/start)

Multi-DM chaining (By Sungwook & Terry)

a) What is the syntax for specifying multiple DMs on one Sapphire object?

Annotation (1st), Nested types (alternative)

b) Any mechanism to detect DM conflicts?

Configuration file that identifies the known good/bad combination

c) What changes do we need to make in code generator?

- Compile time vs Run time
 - 1) Insert the reference to the next DM in the stub code so it is known in compile time.
 - 2) Insert the identifier (maybe a stand-alone DM) that it will use chained DMs.
Application can dynamically inject the DMs.
- Single class DM stub file vs multiple classes DM stub file
 - 1) New DM stub file dictates the chain of references in the code.
 - E.g., while (chain != null) { chain.DM }
 - 2) Multiple classes
- Client stub files: currently there are no client stub files (may not be needed).
- Generate stub files
 - 1) Server/client/group classes of each DM that will have the reference to the next DM to use.
 - 2) Single chained DM stub that will

India team

Srini and Venu