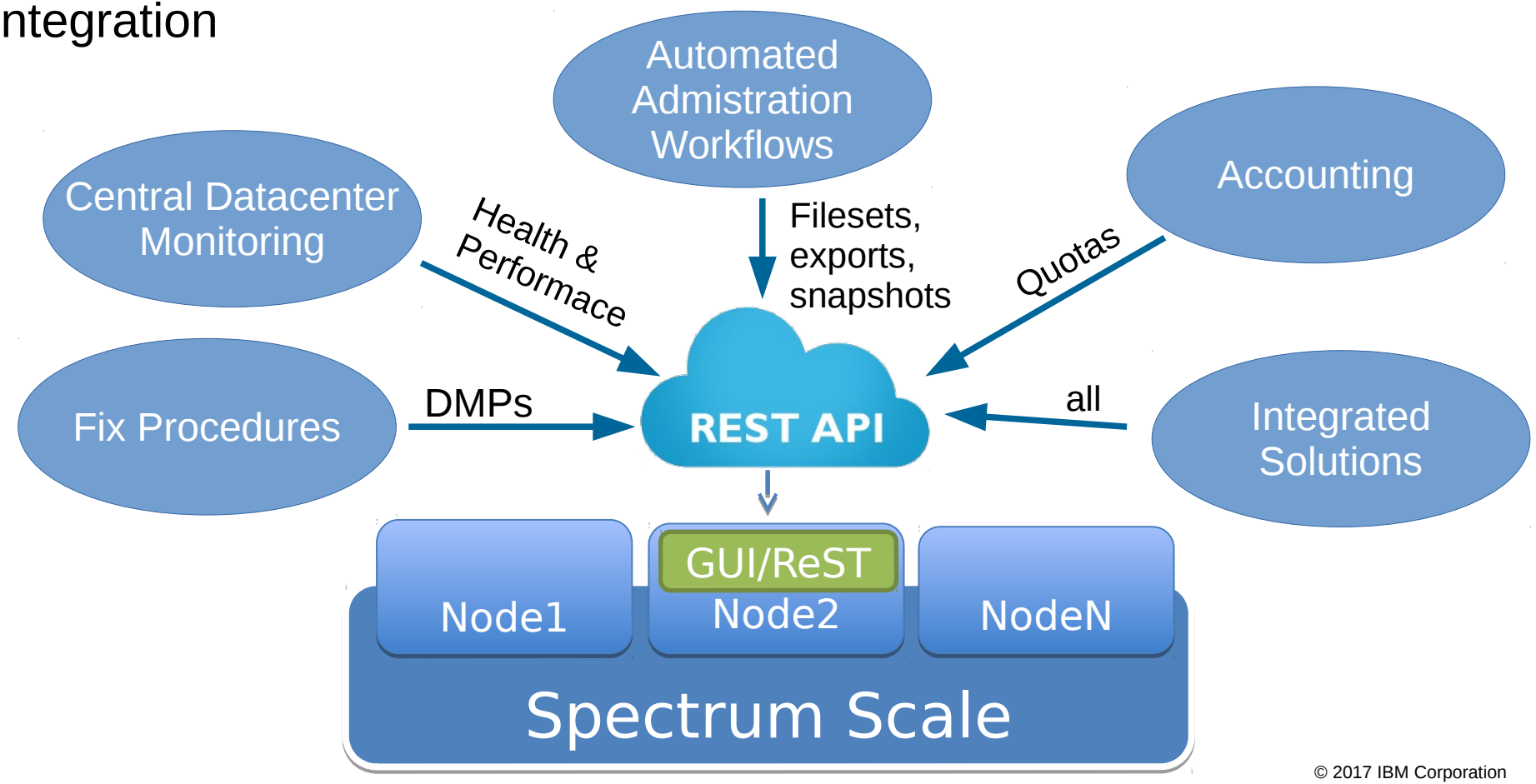
IBM Spectrum Scale | ReST API

ReST API strategic direction

- The ReST API will be THE strategic interface for integrating with 3rd party/customer applications, automation or monitoring.
- Long-term we aim for complete coverage of configuration (mm-commands), health monitoring (mmhealth) and performance monitoring (Zimon).
- ReST API is meant to be scalable and support large clusters (thousands of nodes)
- It runs on a subset of nodes but is highly available (active/active configuration, no ip failover yet, so clients have to switch to alternative ip)
- ReST calls are meant to complete within a few seconds (instead of being open-ended like mm-CLI)

Integration

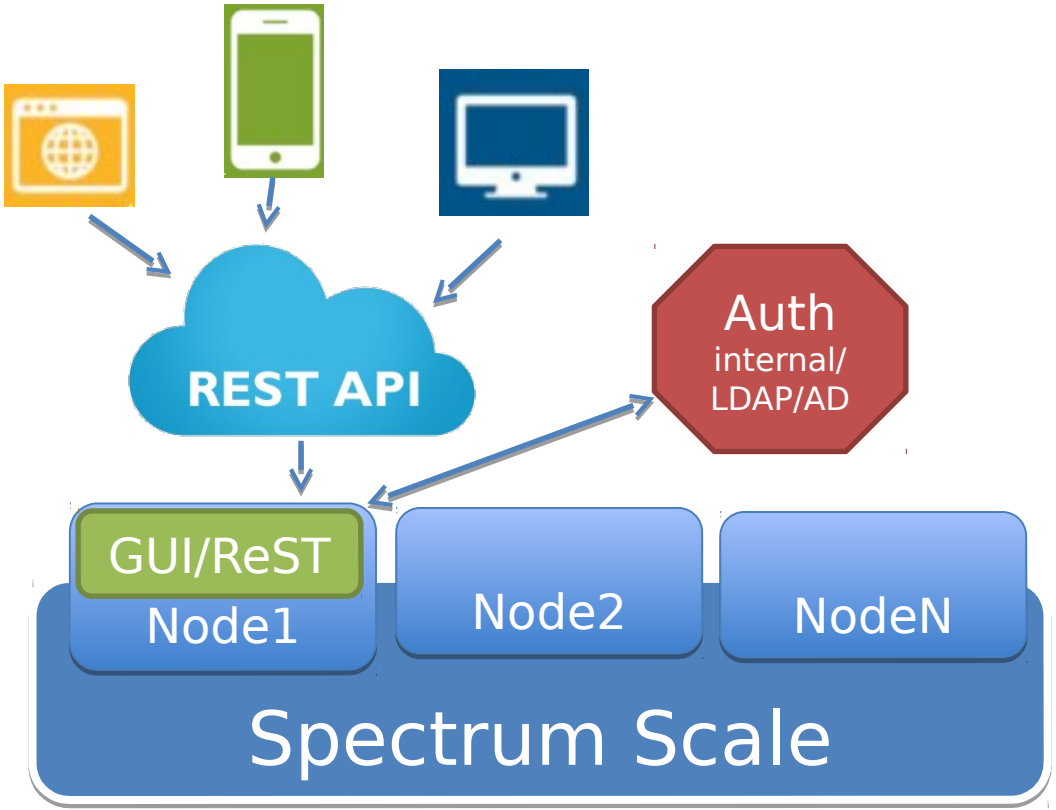


History of ReST API in Spectrum Scale

- Release 4.2.2, V1 API
  - First implementation based on Python
  - Deployment limited to manager nodes, RHEL 7 only
  - Did not meet our expectations and so will be discontinued
- Release 4.2.3, V2 API
  - New implementation based on GUI stack
  - Reuses a lot of the GUI's backend infrastructure. This allows faster coverage of the config model by simultaneously having more mature code
  - Makes deployment easier since it comes with the GUI. No extra service to deploy, no limitations on what nodeclass to deploy
  - Fixes design issues with V1 API in regards of scalability (no paging, field selection, filtering) and command handling (fully async)
  - Extends on data model (NFS & Samba exports, NSDs, health)

REST API architecture (4.2.3)

- In 4.2.3 GUI and ReST API will be driven by the same WebSphere server and share the same object cache.
- Authentication is shared between ReST API and GUI
- The limitation in 4.2.2 that ReST API had to be deployed on manager nodes is gone



ReST API v2

GET	/scalemgmt/v2/ces/addresses	Get listing of CES Addresses
GET	/scalemgmt/v2/ces/addresses/{cesAddress}	Get detailed information about a CES Address
GET	/scalemgmt/v2/ces/services	Get listing of CES Services
GET	/scalemgmt/v2/ces/services/{service}	Get detailed information about a CES Service
GET	/scalemgmt/v2/cluster	Get current configuration information
GET	/scalemgmt/v2/config	Get cluster config
GET	/scalemgmt/v2/filesystems	List of filesystems in the cluster
GET	/scalemgmt/v2/filesystems/{filesystemName}	Get detailed information about a filesystem
GET	/scalemgmt/v2/filesystems/{filesystemName}/acl/{path}	Get access control list of file/directory
PUT	/scalemgmt/v2/filesystems/{filesystemName}/acl/{path}	Write access control list of file/directory

ReST API v2

GET	/scalemgmt/v2/filesystems/{filesystemName}/disks	Get listing of disks
GET	/scalemgmt/v2/filesystems/{filesystemName}/disks/{diskName}	Get detailed information about a disk
GET	/scalemgmt/v2/filesystems/{filesystemName}/filesets	Get listing of filesets
POST	/scalemgmt/v2/filesystems/{filesystemName}/filesets	Create a new fileset
DELETE	/scalemgmt/v2/filesystems/{filesystemName}/filesets/{filesetName}	Delete a fileset
GET	/scalemgmt/v2/filesystems/{filesystemName}/filesets/{filesetName}	Get detailed information about a fileset
PUT	/scalemgmt/v2/filesystems/{filesystemName}/filesets/{filesetName}	Change an existing fileset
DELETE	/scalemgmt/v2/filesystems/{filesystemName}/filesets/{filesetName}/link	Unlink an existing fileset
POST	/scalemgmt/v2/filesystems/{filesystemName}/filesets/{filesetName}/link	Link an existing fileset

ReST API v2

GET	/scalemgmt/v2/filesystems/{filesystemName}/filesets/{filesetName}/quotas	List quotas in the cluster
POST	/scalemgmt/v2/filesystems/{filesystemName}/filesets/{filesetName}/quotas	Set quota limits / default quota limits / quota grace settings
GET	/scalemgmt/v2/filesystems/{filesystemName}/filesets/{filesetName}/snapshots	List snapshots for the specified fileset
POST	/scalemgmt/v2/filesystems/{filesystemName}/filesets/{filesetName}/snapshots	Create a new snapshot
DELETE	/scalemgmt/v2/filesystems/{filesystemName}/filesets/{filesetName}/snapshots/{snapshotName}	Delete a snapshot
GET	/scalemgmt/v2/filesystems/{filesystemName}/filesets/{filesetName}/snapshots/{snapshotName}	Read a snapshot for a fileset
GET	/scalemgmt/v2/filesystems/{filesystemName}/owner/{path}	Get file/directory owner
PUT	/scalemgmt/v2/filesystems/{filesystemName}/owner/{path}	Set file/directory owner
GET	/scalemgmt/v2/filesystems/{filesystemName}/quotas	List quotas in the cluster
POST	/scalemgmt/v2/filesystems/{filesystemName}/quotas	Set quota limits / default quota limits / quota grace settings

ReST API v2

GET	/scalemgmt/v2/filesystems/{filesystemName}/snapshots	List snapshots in a file system
POST	/scalemgmt/v2/filesystems/{filesystemName}/snapshots	Create a new snapshot
DELETE	/scalemgmt/v2/filesystems/{filesystemName}/snapshots/{snapshotName}	Delete a snapshot
GET	/scalemgmt/v2/filesystems/{filesystemName}/snapshots/{snapshotName}	Read a snapshot for a fileset
GET	/scalemgmt/v2/info	Get REST API info
GET	/scalemgmt/v2/jobs	Get list of asynchronous jobs
GET	/scalemgmt/v2/jobs/{jobId}	Get job details
GET	/scalemgmt/v2/nfs/exports	List of export in the cluster
POST	/scalemgmt/v2/nfs/exports	Create a new nfs export
GET	/scalemgmt/v2/nfs/exports/{exportPath}	Get detailed information about a export
PUT	/scalemgmt/v2/nfs/exports/{exportPath}	Change an existing nfs export
DELETE	/scalemgmt/v2/nfs/exports/{path}	Delete a NFS export

ReST API v2

GET	/scalemgmt/v2/nodes	Get listing of Nodes
GET	/scalemgmt/v2/nodes/{name}	Get listing for a Node
GET	/scalemgmt/v2/nodes/{name}/health/events	Get System Health events
GET	/scalemgmt/v2/nodes/{name}/health/states	Get System Health events
GET	/scalemgmt/v2/nsds	Get listing of nsds
GET	/scalemgmt/v2/nsds/{nsdName}	Get detailed information about a nsd
GET	/scalemgmt/v2/smb/shares	List of smb shares in the cluster
POST	/scalemgmt/v2/smb/shares	Create a new smb share
DELETE	/scalemgmt/v2/smb/shares/{shareName}	Delete a SMB share
GET	/scalemgmt/v2/smb/shares/{shareName}	Get detailed information about a smb shares
PUT	/scalemgmt/v2/smb/shares/{shareName}	Change an existing smb share

Paging

- Since a GET might query for thousands of objects we need a mechanism to avoid responses to grow towards the gigabyte range.
- Therefore the GET will return only up to 1000 objects per single request. More objects can be retrieved with subsequent GET requests.
- Response objects may contain an object called "paging" which contains a link which can then be used to retrieve the next set of objects (if available), e.g.:

```
"paging" : {
      "next" : "/filesystems?filter=filesystemName='''gpfs.*'''&lastId=12"
}
```

Fields

The "field" parameter can be used to specify the fields of the returned object by adding the query parameter "fields" to a GET URL, e.g. "?fields=field1,field2,parentField1.field1".

There are two special field names:

- _none_: Will only return the fields needed to uniquely identify an object (eg filesystemName and filesetName for a fileset)
- _all_: Will return all fields

The default for queries that return more than one object depends on the specific endpoint. We return whatever we think are the fields that someone would like to see in a listing. The default for queries to a single object is _all_.

To query fields that are wrapped in a parent object a "." can be used. For example to query for "iamMode" in the object below "?fields=config.iamMode" can be used:

To query for all fields of a parent the parent name only can be used, e.g. "?fields=config,afm" will include all fields of the parent objects "config" and "afm"

Multiple fields can be separated using ",", e.g. "?fields=config.iamMode,state.id"

Example: `https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?fields=status.rootInode,status.id`

Filter

The filter parameter can be used to filter the retrieved objects. For example all filesets with id<2 can be retrieved by using "?filter=state.id<2". The fields used in the filter parameter are always added to the result set and therefore don’t have to be specified using the "fields" mechanism. Filters can be concatenated using",". The supported operators for different field types are as follows:

- String: "=","!=" using a regular expression as the value is supported, e.g. "?filter=config.filesetName=' ' 'fs.*' ' '"
- If a regex is used it has to start with ' ' ' and end with ' ' '
- Numeric: "=", "!=", "<", ">" e.g. "?filter=status.id<2"
- Boolean: "=","!=" using "true" or "false" as the value, e.g. "?filter=afm.afmEnableAutoEviction=true"

Example: `https://localhost:443/scalemgmt/v2/filesystems/gpfs0/filesets?filter=status.rootInode>3`

Async execution

- The mm-command line does not provide any guarantees on command runtime. This means that basically every command might take any amount of time to complete. Even commands like mmcrsnapshot that usually complete inside a few milliseconds may take many minutes in certain circumstances.
- Therefore in v2 all POST, PUT and DELETE request will be async. As response the client will get a job object it can query for the state of the operation


```
{
  "jobs" : [ {
    "jobId" : 1,
    "submitted" : "2017-02-23 11.34.29"
    "completed" : "N/A",
    "status" : "RUNNING",
    "request" : {
      "data" : {
        "config" : {
          "path" : "/mnt/gpfs0/smbexport04",
          "shareName" : "smbexport04"
        },
        "smbOptions" : {
          "browseable" : "yes",
          "comment" : "!@#%^&*()_+`~<>:\\\"{}[];',.?/|\\%3A0x00",
          "gpfs:sharemodes" : "no"
        }
      },
      "type" : "POST",
      "url" : "/scalemgmt/v2/smb/shares"
    },
    "result" : { },
  } ],
  "status" : {
    "code" : 202,
    "message" : "The request was accepted for processing"
  }
}
```



```
{
  "jobs" : [ {
    "jobId" : 4,
    "submitted" : "2017-02-23 11.41.10"
    "completed" : "2017-02-23 11.42.08",
    "status" : "COMPLETED",
    "request" : {
      "data" : {
        "config" : {
          "path" : "/mnt/gpfs0/smbexport04",
          "shareName" : "smbexport05"
        },
        "smbOptions" : {
          "browseable" : "yes",
          "comment" : "!@#%^&*()_+~<>:\\{}[];,.?/|\\%3A0x00",
          "gpfs:sharemodes" : "no"
        }
      },
      "type" : "POST",
      "url" : "/scalemgmt/v2/smb/shares"
    },
    "result" : {
      "commands" : [ "mmsmb export add 'smbexport05' '/mnt/gpfs0/smbexport04'
--option 'browseable=yes' --option 'gpfs:sharemodes=no' --option 'comment=!@#
%^&*()_+~<>:\\{}[];,.?/|\\%3A0x00' " ],
      "exitCode" : 0,
      "progress" : [ ],
      "stderr" : [ ],
      "stdout" : [ "info: mmsmb export add: The SMB export was created
successfully.\n" ]
    },
  } ],
  "status" : {
    "code" : 200,
    "message" : "The request finished successfully"
  }
}
```

Roadmap

- Add ReST for Zimon data.
- Use ReST for our own GUI frontend in case of AFM
- Allow to cancel jobs (at least those in which case the mm-command supports that)
- More granular authorization (CRUD per endpoint) for REST as alternative to the current role concept
- Node management: Nodeclasses, add node, remove node
- More PUT, POST, DELETE support throughout the model
- More coverage in general with priority on new features
- Push API using WebSockets or something similar for health events, performance, config changes