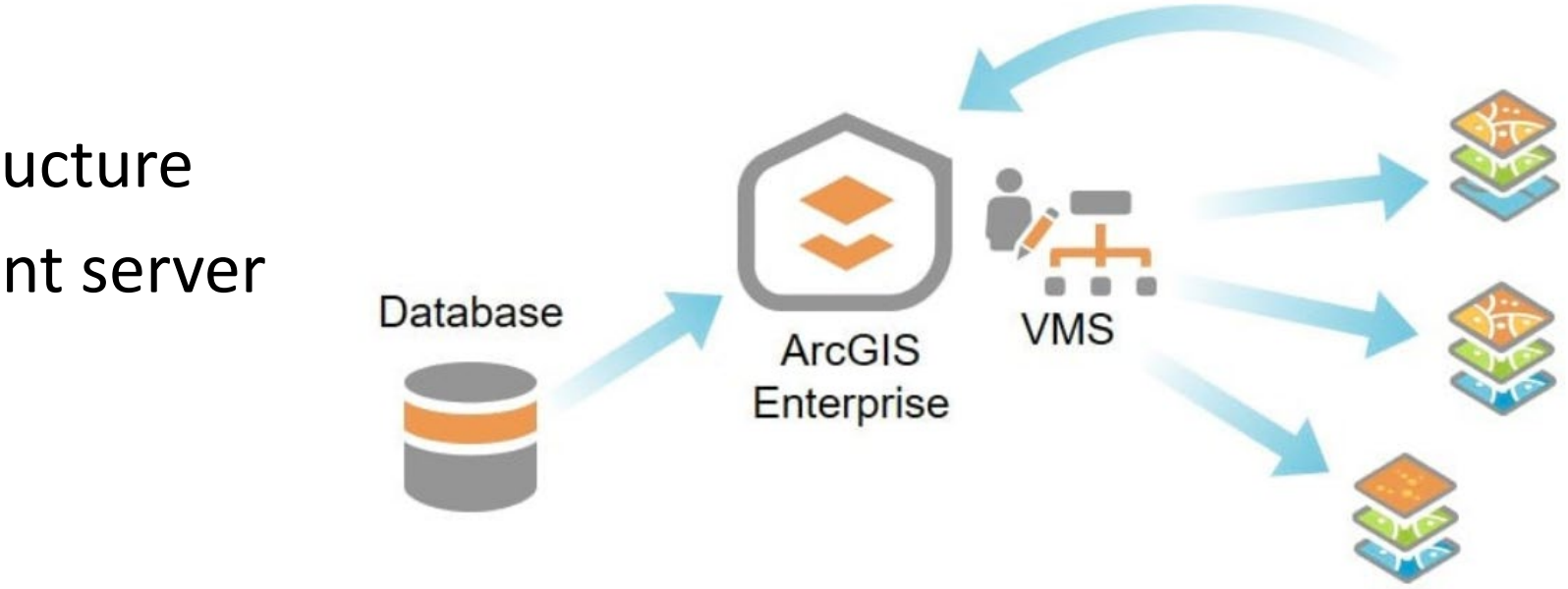
Version Management with ArcGIS

Annie Sasidar | Kevin Watt | Erik Hoel

Agenda

- Overview
- Table and query structure
- Version management server
  - REST API
- Pro managed SDK



What is branch versioning?

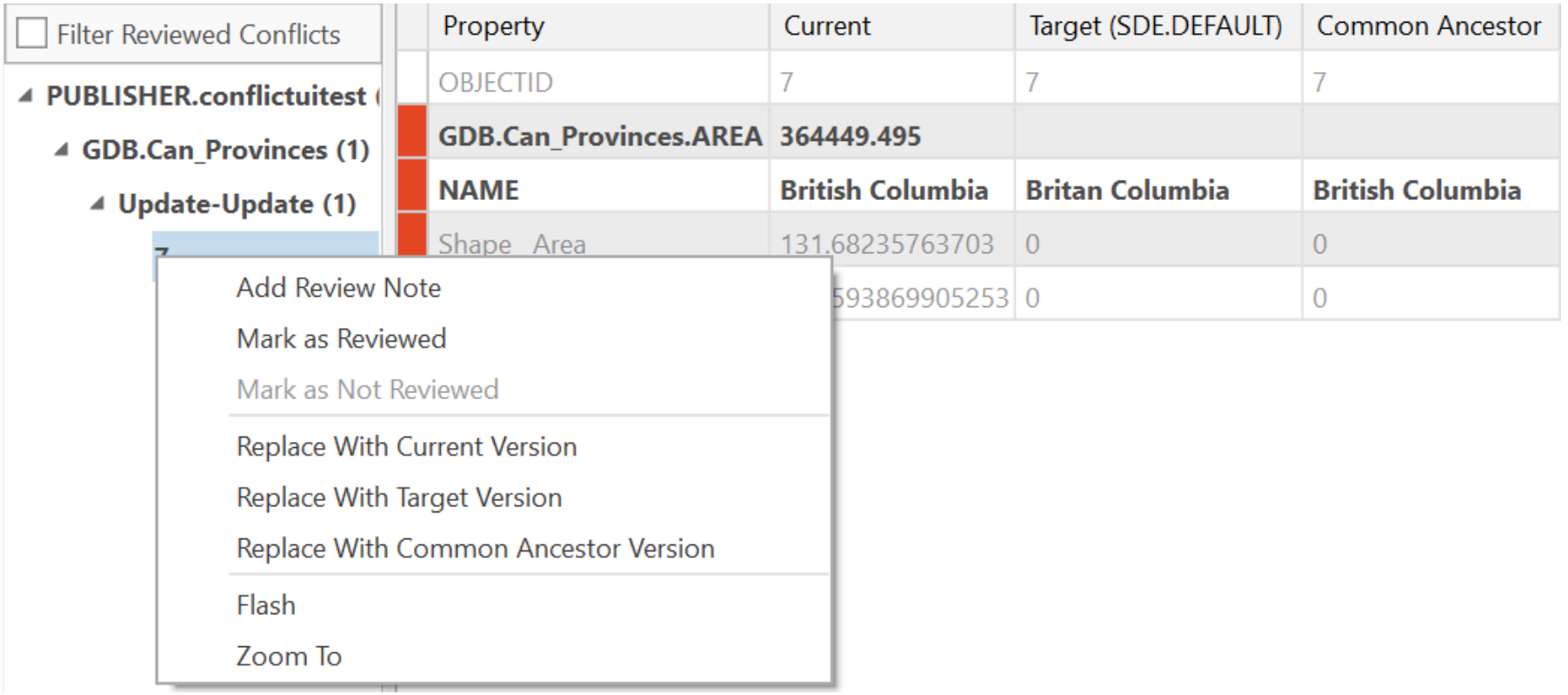
- esri's new versioning model leveraging a services-based architecture
- Builds on the well-established capabilities of traditional versioning
  - Multi-user editing
  - Edit isolation in child versions
- The Version Management Server (VMS) supports key version administration tasks through services
- Branch versioned datasets are read-only in client-server (direct connect)

Benefits

- Performance:
  - Archive edits are written to a single table
  - Version queries are simpler and more performant
- Administration is simpler
  - No need to reconcile versions to DEFAULT to achieve a full compress
  - No compress
- Usability - ability to directly edit a version
  - DEFAULT in Pro
  - Any version via the rest end point, w/o an “edit session”
- Multiple versions can post in parallel
- Archiving is built-in
  - Track deleted features

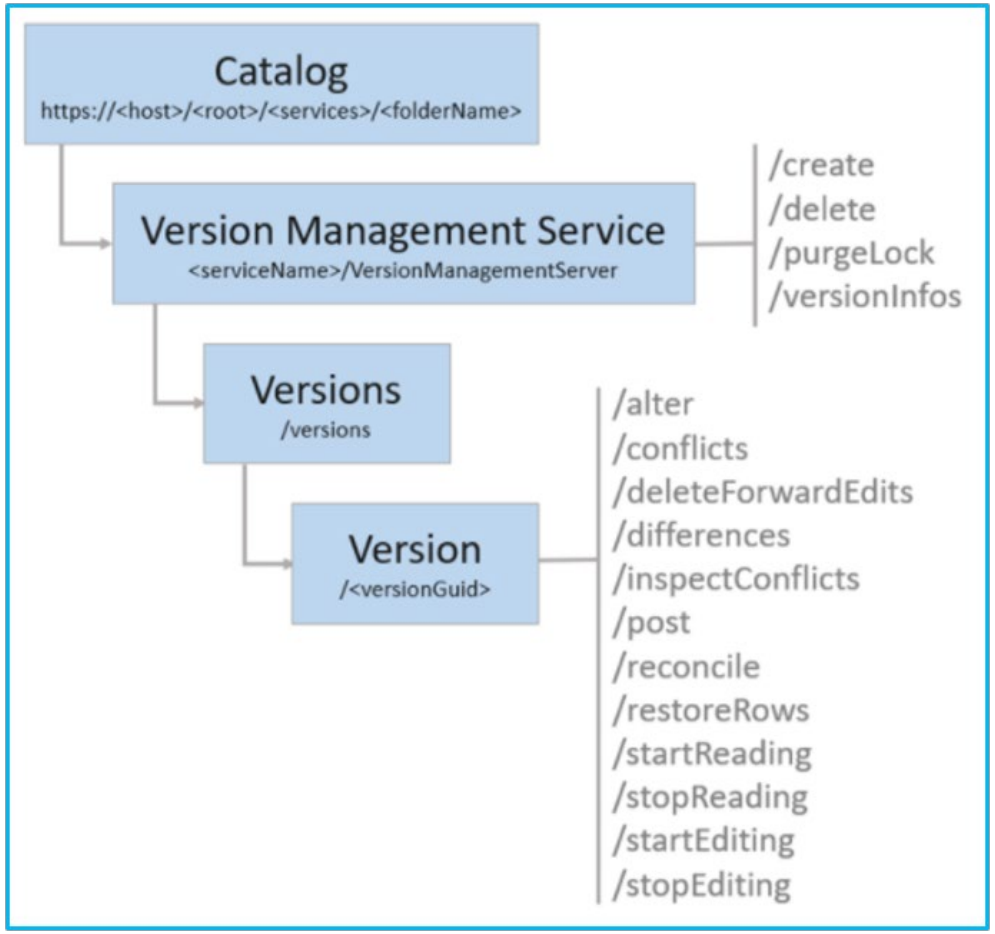
Benefits

- Conflict management is enhanced to persist across multiple editing sessions
- Ability to add notes/comments about conflicts during QA



Benefits

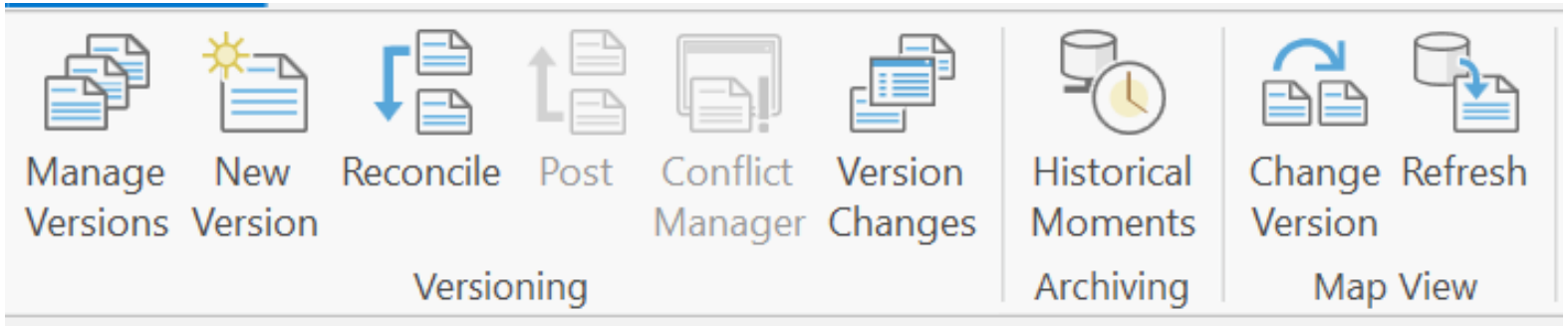
- REST end point access to the version management operations



`https://<catalog-url>/<serviceName>/VersionManagementServer`

Version model similarities

- ArcGIS Pro UI/UX
  - Version Manager
  - Versioning Ribbon
  - GP tools



- Versioning operations
  - Create/ Alter / Delete version
  - Access type
  - Undo/Redo and edit isolation in child versions
  - Reconcile
  - Detect and Resolve conflicts
  - Post

Version model differences

- Create:
  - Version management only available through services
  - Grand child versions not supported
- Edit:
  - Only available through services; branch versioned datasets are read-only in client-server
  - Insert-only model - this allows us to track changes over time including deletes
  - DEFAULT version does not support Undo/Redo
  - Child versions support only one editor or multiple readers at a time
- QA:
  - Reconcile is not undoable
  - Conflicts can be persisted across multiple edit sessions
  - Version ownership can be altered to support QA workflows

Version model differences

- Support:
  - Available throughout the platform
- Solutions:
  - Utility network, parcel management, roads and highways

Minimum setup configuration

ArcGIS Enterprise 10.6

- SQL Server 2012 SP3

- 9.5

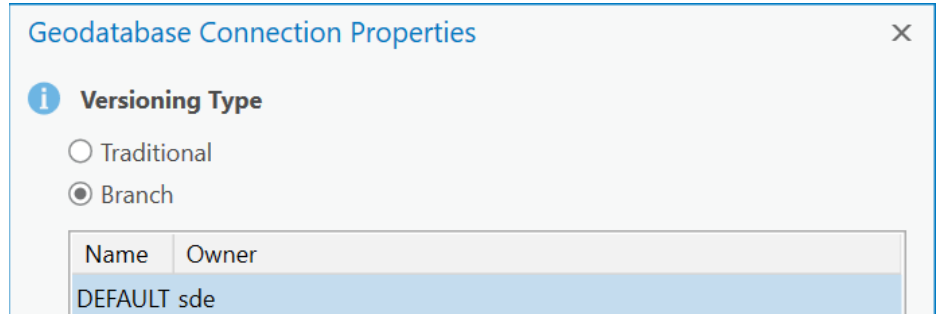
- 10.5 FP5

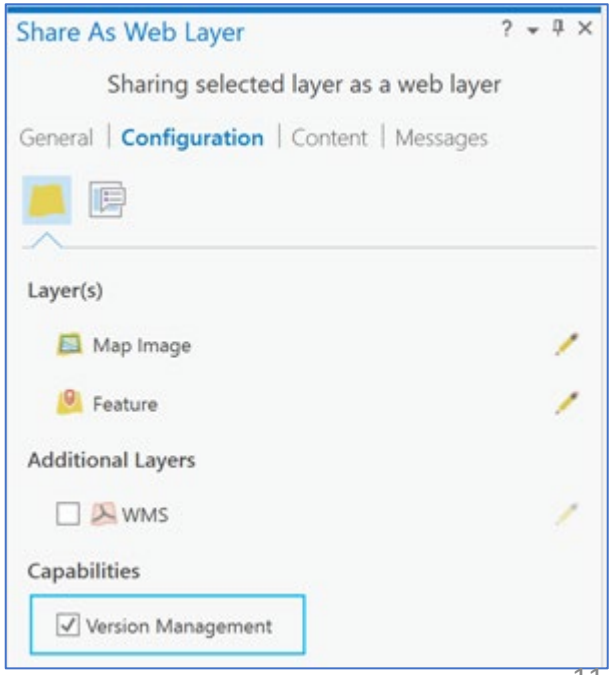
- 12.1.0.2

- SAP HANA 2

Setting up for branch versioning

- Set workspace connection to use branch versioning
- Add a Global ID
- Enable editor tracking
- Register data as versioned
- Connect to a portal as a user with publisher privileges
- Share as web layer and select VMS capability





Enhancements

- Pro 2.6 / Server 10.8.1
  - Column level conflict detection
  - Filter fields in conflict detection
  - Fine grained Enterprise privilege to perform version administration
- Pro 2.7 / Server 10.9
  - Difference between moments on DEFAULT version
  - Partial Post by selection set

Behind the scenes – database queries

Kevin Watt

Data Model

OBJECTID	SHAPE
1	. . .

OBJECTID	SHAPE	GDB_BRANCH_ID	GDB_FROM_DATE	GDB_IS_DELETE	GDB_ARCHIVE_OID
1	. . .	0	FEB-05-20 14:34	0	1

GDB_BRANCH_ID - (INT64) Version identifier to isolate edits

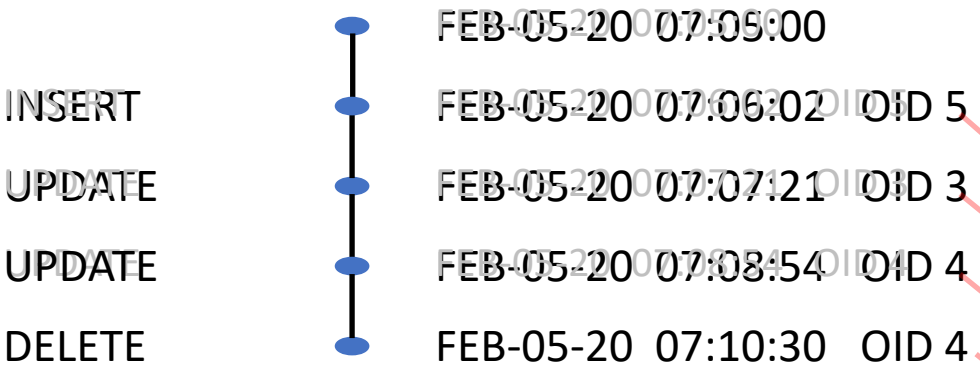
GDB_FROM_DATE - (TIMESTAMP) field to record edit moments in fractional seconds

GDB_IS_DELETE - (CHAR(1)) field to record whether the edit is a deletion

GDB_ARCHIVE_OID - (INT64) unique row identifier

Fields are managed internally and are available when working with the Archive class

BRANCH 0



OBJECTID	BRANCH_ID	GDB_FROM_DATE	IS_DELETE
1	0	FEB-05-20 07:05:00	0
2	0	FEB-05-20 07:05:00	0
3	0	FEB-05-20 07:05:00	0
4	0	FEB-05-20 07:05:00	0
5	0	FEB-05-20 07:06:02	0
3	0	FEB-05-20 07:07:21	0
4	0	FEB-05-20 07:08:54	0
4	0	FEB-05-20 07:10:30	1

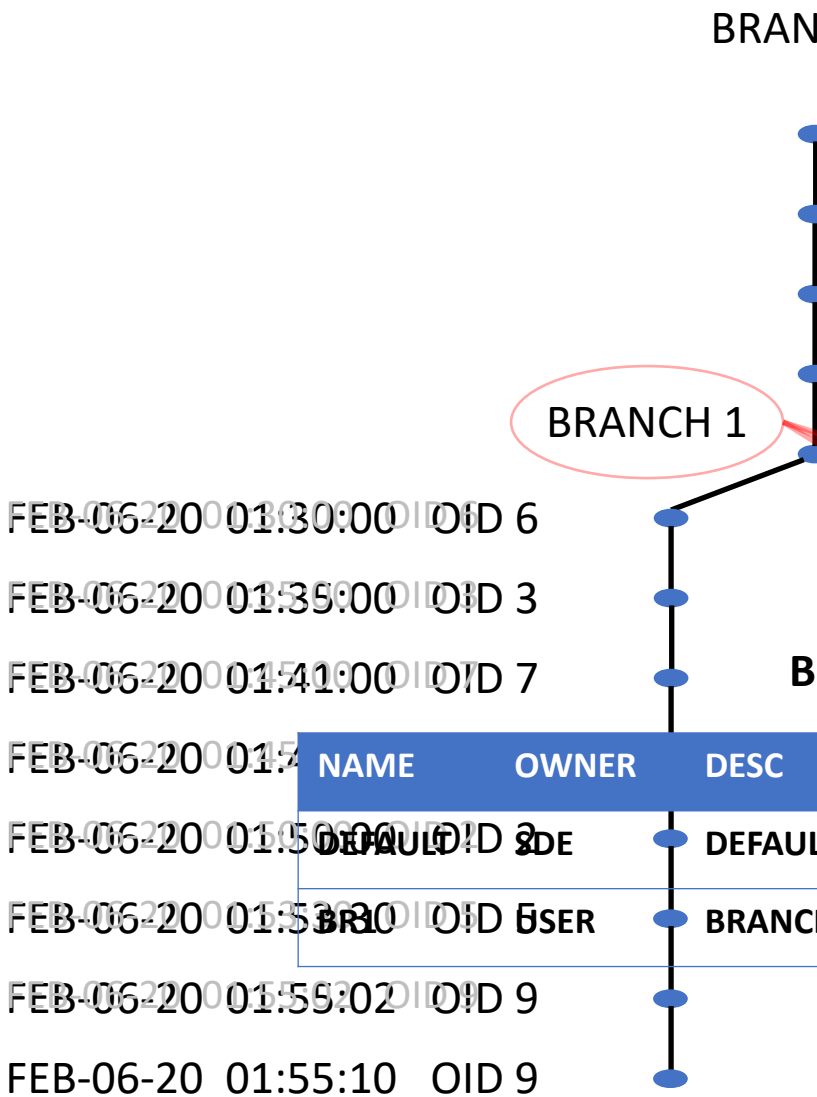
Current Moment

```
1  SELECT OBJECTID
   FROM DISTRIBUTIONDEVICE
   WHERE DISTRIBUTIONDEVICE.rowid IN
2    (SELECT MB_.RID
     FROM
3       (SELECT ROWID,
            ROW_NUMBER() OVER (PARTITION BY objectid
            ORDER BY gdb_from_date DESC) rn, gdb_is_delete
        FROM DISTRIBUTIONDEVICE
        WHERE (gdb_branch_id = 0)) MB_
4    WHERE rn = 1 AND gdb_is_delete = '0');
```

RN	OBJECTID	BRANCH_ID	GDB_FROM_DATE	IS_DELETE
1	1	0	FEB-05-20 07:05:00	0
1	2	0	FEB-05-20 07:05:00	0
2	3	0	FEB-05-20 07:05:00	0
1	5	0	FEB-05-20 07:06:02	0
3	4	0	FEB-05-20 07:06:02	0
2	4	0	FEB-05-20 07:08:54	0
1	4	0	FEB-05-20 07:08:54	1
1	5	0	FEB-05-20 07:06:32	0
	6	10	FEB-06-20 11:27:10	0

Current moment result set

Order by GDB_FROM_DATE and assign ROW_NUMBER() RN to each row



OBJECTID	BRANCH_ID	GDB_FROM_DATE	IS_DELETE
1 . . .	0 . . .	FEB-05-20 07:05:00	0 . . .
4	0	FEB-05-20 07:08:50	0
8	0	FEB-05-20 07:10:30	0
6	0	FEB-06-20 01:30:00	0
3	0	FEB-06-20 01:35:00	0
7	0	FEB-06-20 01:41:00	0
4	0	FEB-06-20 01:48:00	0
2	0	FEB-06-20 01:50:00	1
5	1	FEB-06-20 01:53:30	0

BRANCHES

NAME	OWNER	DESC	STATUS	BRANCH_ID	CREATION_TIME	BRANCH_MOMENT	ANCESTOR_MOMENT
DEFAULT	SDE	DEFAULT	PUBLIC	0	JUL-13-19 07:05	FEB-06-20 01:55:10	1
BR1	USER	BRANCH 1	PUBLIC	1	FEB-06-20 01:25	FEB-06-20 01:25	FEB-06-20 01:25

Current Branch Moment Query

```
SELECT OBJECTID
FROM DISTRIBUTIONDEVICE
WHERE DISTRIBUTIONDEVICE.rowid IN
  (SELECT MB_.RID
   FROM
    (SELECT ROWID,
       ROW_NUMBER() OVER (PARTITION BY objectid
       ORDER BY gdb_from_date DESC) rn,
                gdb_is_delete
     FROM DISTRIBUTIONDEVICE
     WHERE ((gdb_branch_id = 0 AND
              gdb_from_date < :ancector_moment) OR
            (gdb_branch_id = :branch_id AND
              gdb_from_date <= :specific_moment)) MB_

   WHERE rn = 1 AND gdb_is_delete = '0');
```

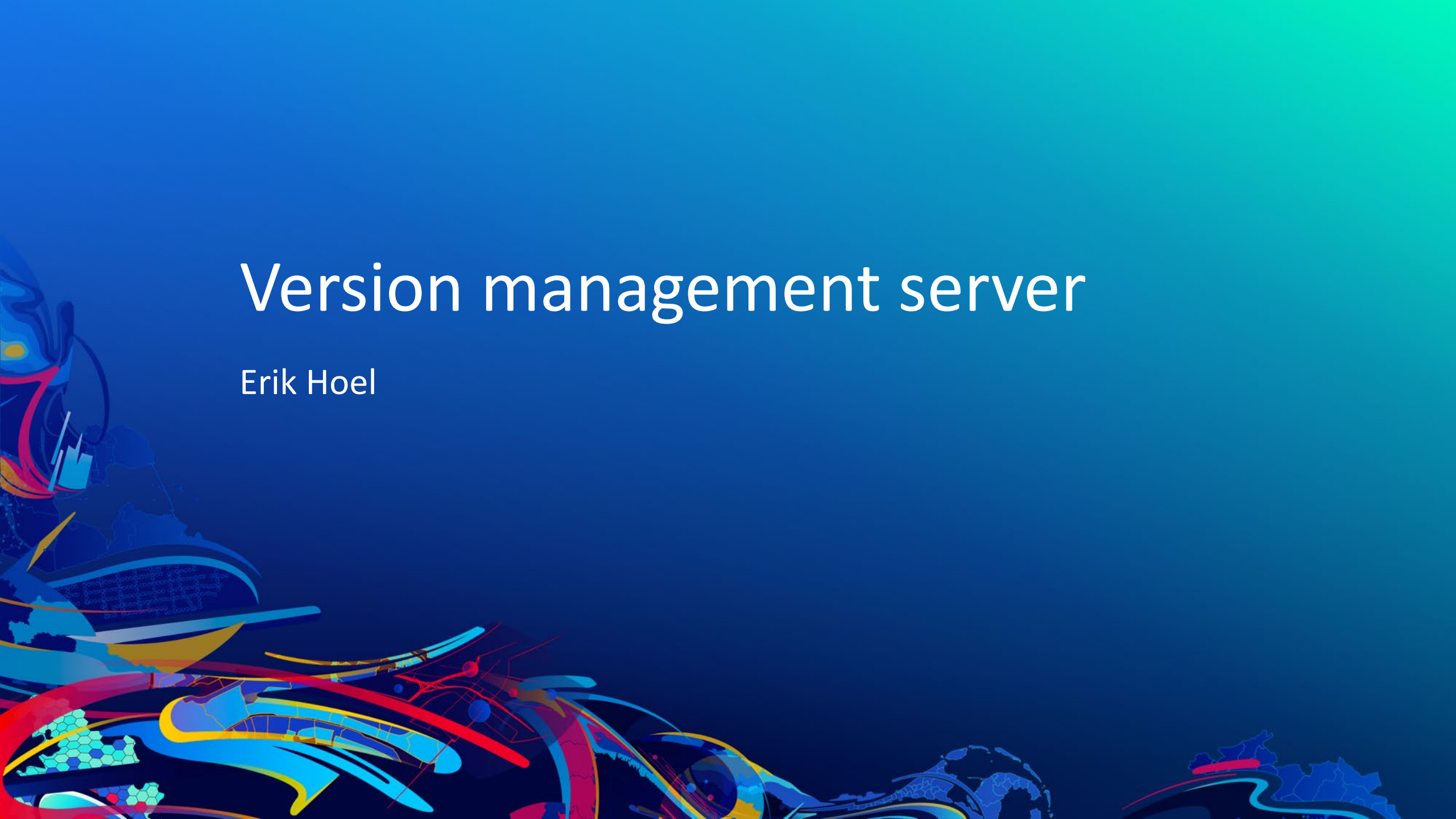
OBJECTID	BRANCH_ID	GDB_FROM_DATE	IS_DELETE
1	0	FEB-05-20 07:05:00	0
3	1	FEB-06-20 01:35:00	0
5	1	FEB-06-20 01:53:30	0
6	1	FEB-06-20 01:30:00	0
7	1	FEB-06-20 01:45:00	0
3	0	FEB-05-20 07:07:21	0
4	0	FEB-05-20 07:08:54	0
4	0	FEB-05-20 07:10:30	1
6	1	FEB-06-20 01:30:00	0
3	1	FEB-06-20 01:35:00	0
7	1	FEB-06-20 01:41:00	0
7	1	FEB-06-20 01:45:00	0
2	1	FEB-06-20 01:50:00	1
5	1	FEB-06-20 01:53:30	0
9	1	FEB-06-20 01:55:02	0
9	1	FEB-06-20 01:55:10	1

Current Branch Moment result set

OBJECTID 7 and OBJECTID 9 are filtered from Branch 1

Version management server

Erik Hoel

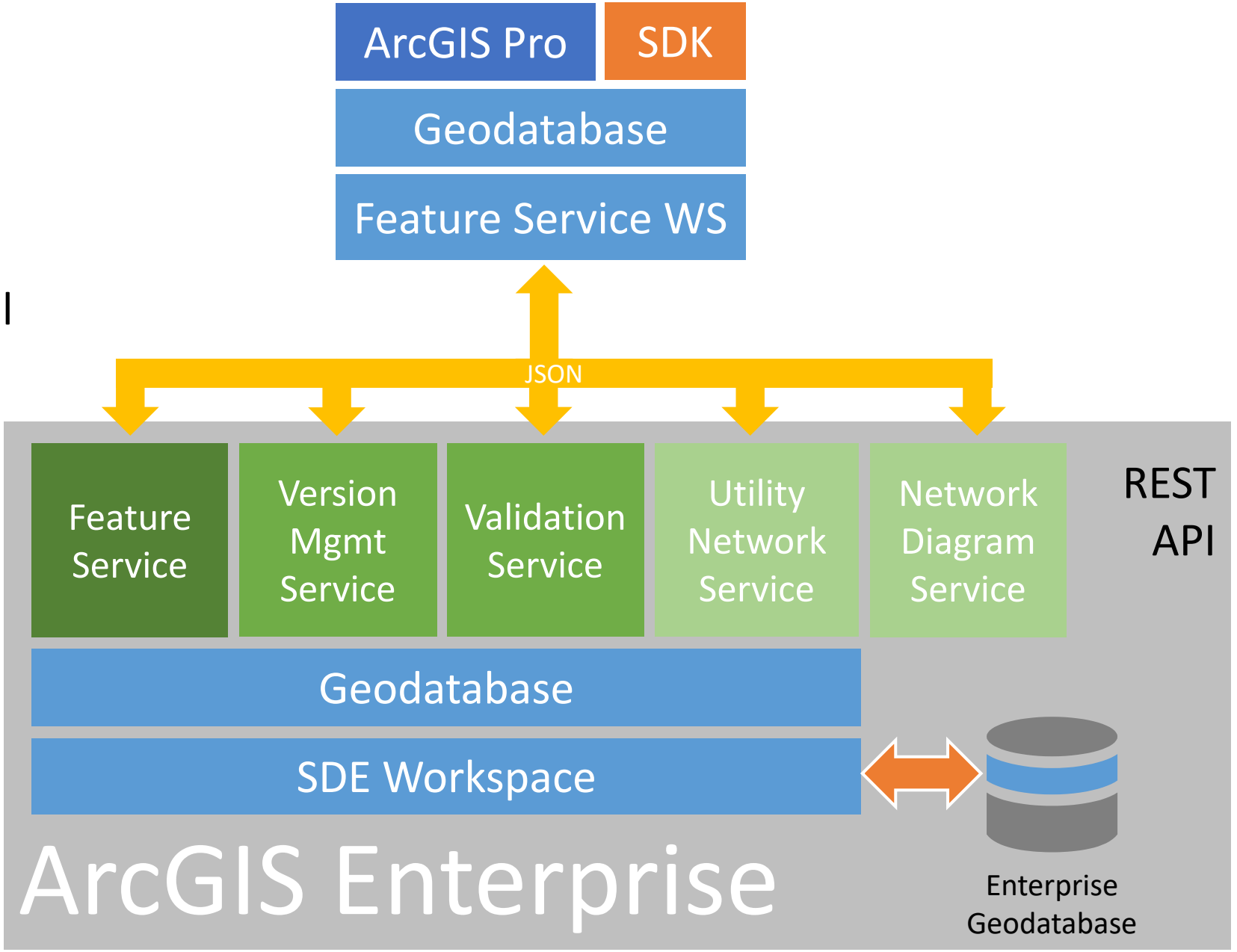


Architecture

overview

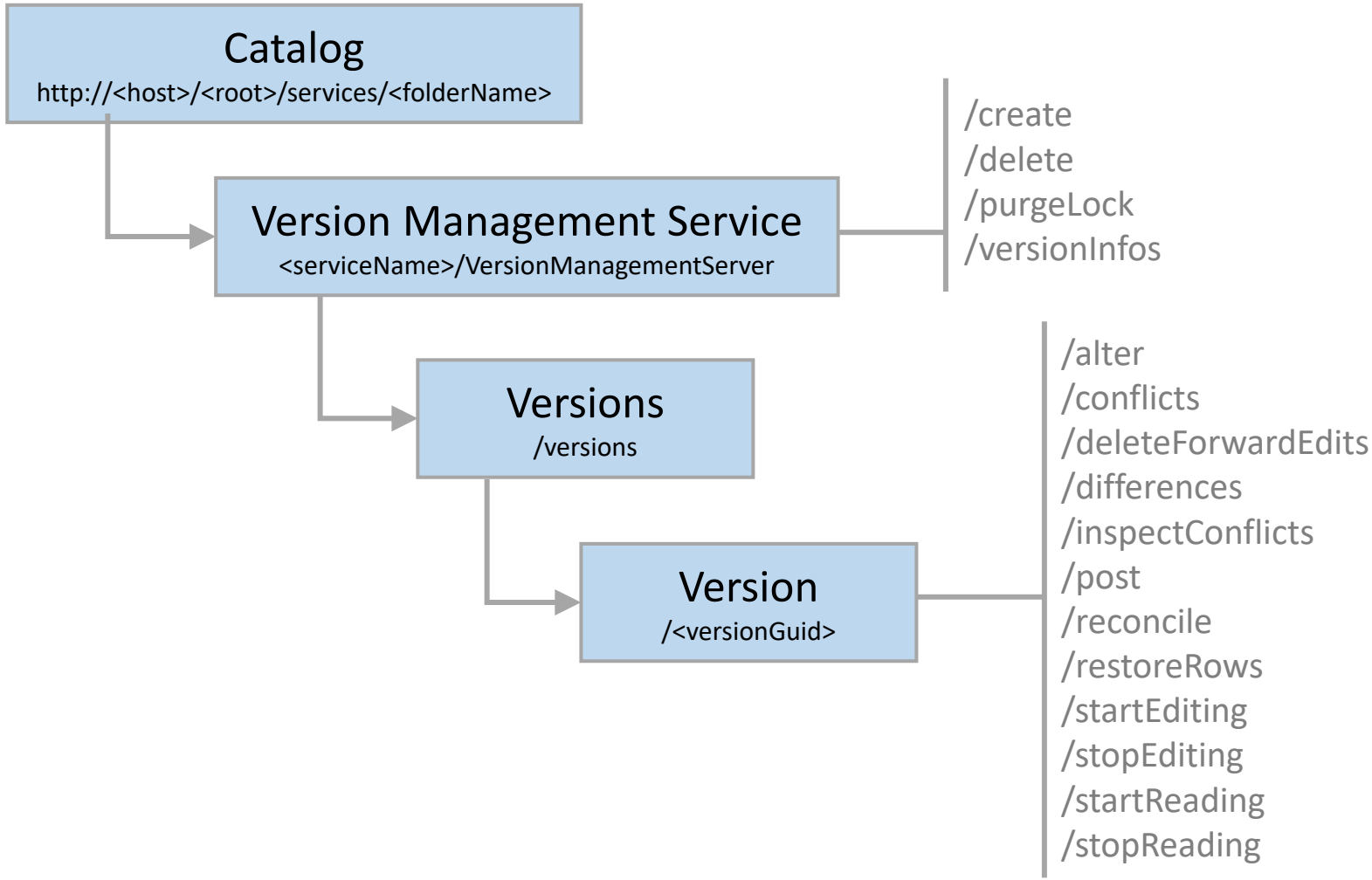
The version management server exposes managerial capabilities necessary to support feature services when working with branch versioned dataset; e.g.,

- Creating and deleting versions
- Controlling service sessions
- Reconcile and post
- Managing conflicts



Version management server

REST API

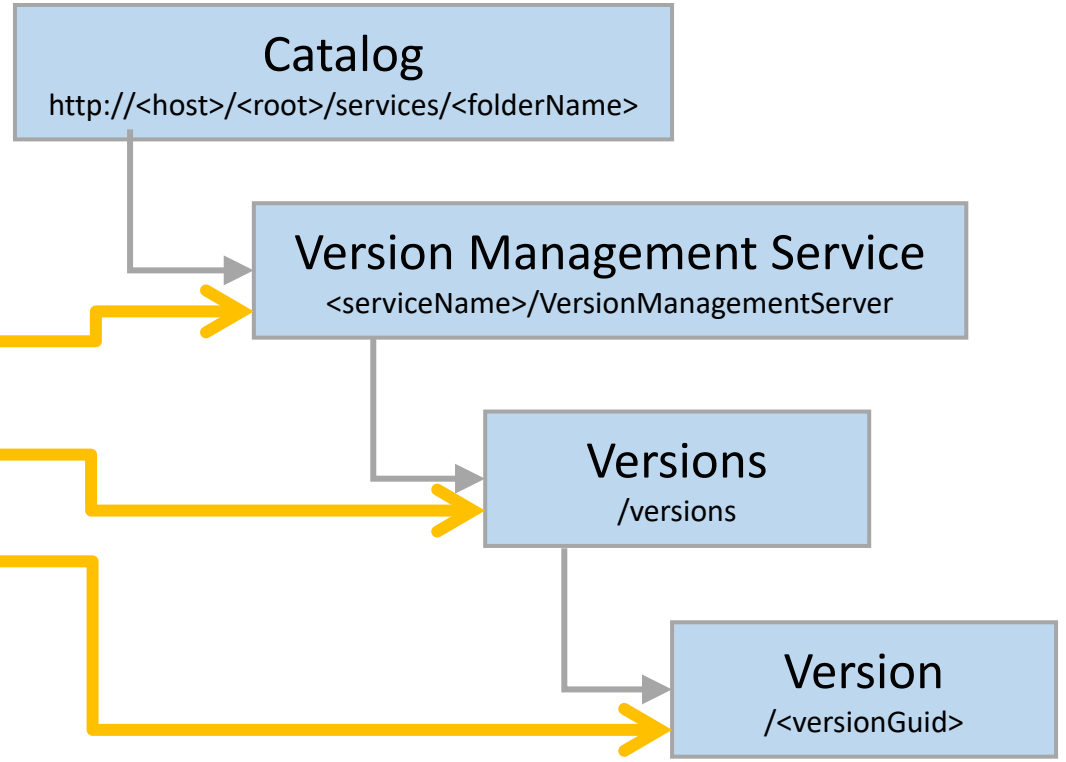


Resources

root

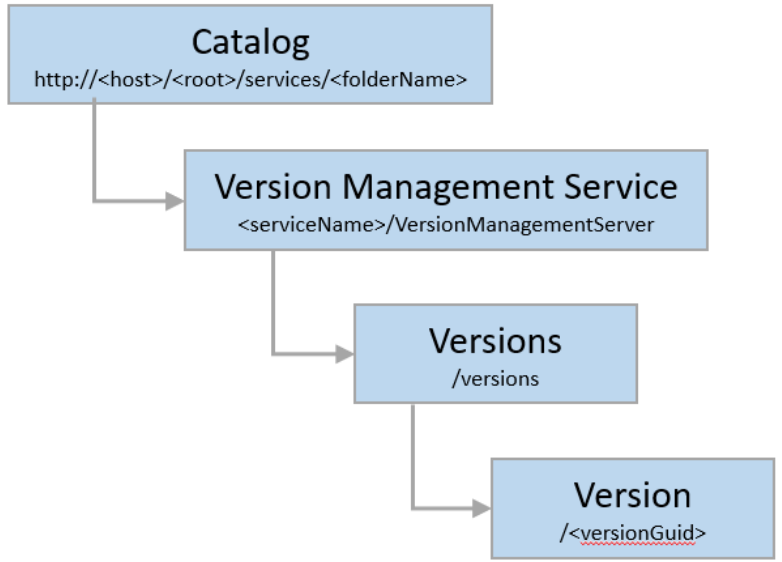
root/versions

root/versions/<versionGuid>



root

resource response



root

JSON Response:

```
{
  “name” : “Version Management Server”,
  “type” : “Map Server Extension”,
  “defaultVersionName” : <string>,
  “defaultVersionGuid” : <guid>,
  “capabilities” : {
    “supportsConflictDetectionByAttribute” : <boolean>, // 10.8.1
    “supportsHistoricalMarkers” : <boolean>,            // 10.8.1
    “supportsPartialPost” : <boolean>,                  // 10.9
    “supportsDifferencesFromMoment” : <boolean>,        // 10.9
    “supportsDiagnostics” : <boolean>
  }
}
```

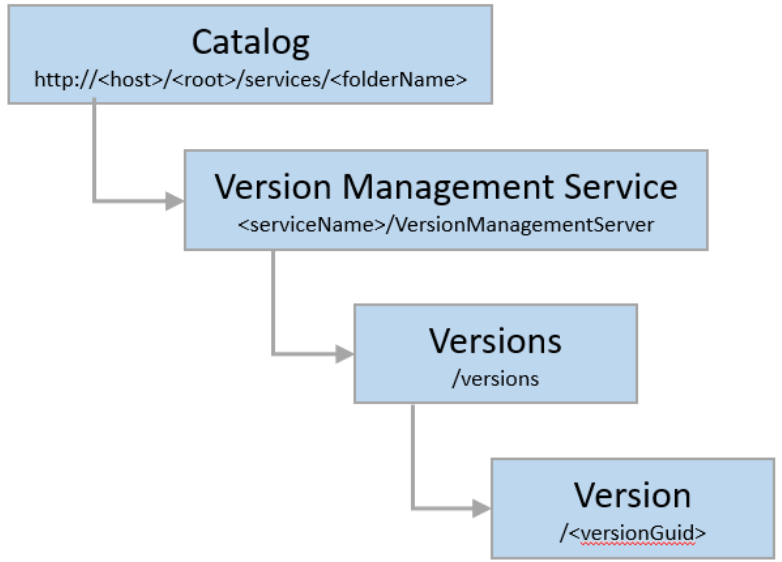
root

f=schema

root?f=schema

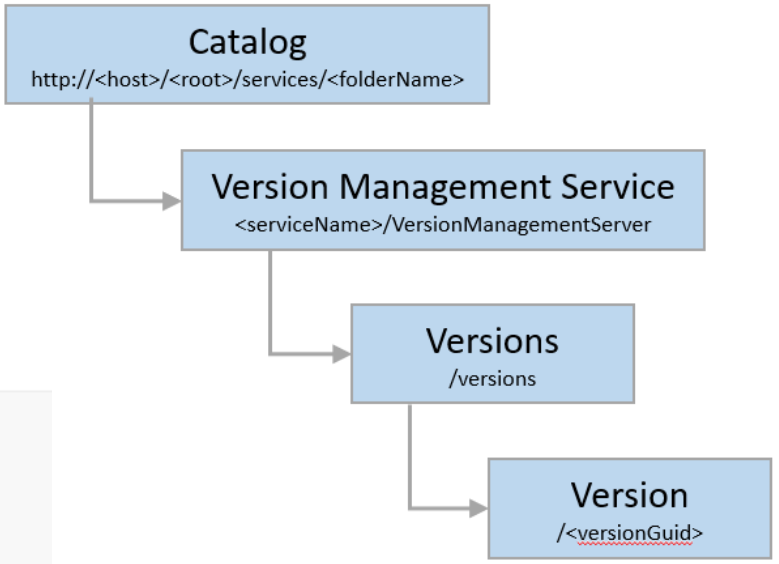
JSON Response Syntax:

```
{
  “name” : “”,
  “operations” : [
    {
      “name” : “<operationName>”,
      “parameters” : [ “<parameterName>” ],
      “supportedOutputFormats” : [ “<outputFormat>” ],
      “requiredCapability”: “<requiredCapability>”
    }
  ],
  “resources” : [
    {
      “name” : “<resourceName>”,
      “isCollection” : <true | false>,
      “isStatic” : <true | false>,
      “operations” : [
        {
          “name” : “<operationName>”,
          “parameters” : [ “<parameterName>” ],
          “supportedOutputFormats” : [ “<outputFormat>” ],
          “requiredCapability” : “<requiredCapability>”
        }
      ]
    }
  ],
  “schemaVersion” : <schemaVersion>
}
```



root/versions

resource response



JSON Response Syntax

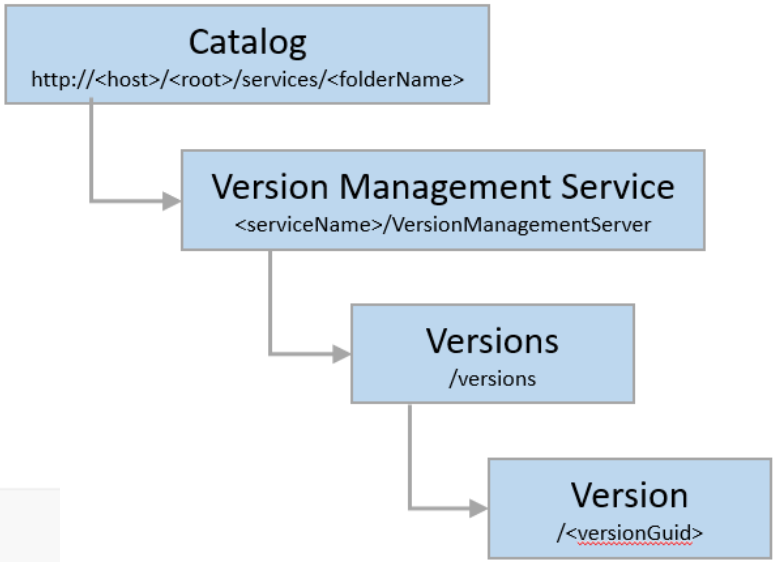
```
{
  "versions" : [
    {
      "versionName": "<versionName>",
      "versionGuid": <guid>
    }
  ]
}
```

JSON Response Example

```
{
 "versions": [
  {
   "versionName": "SDE.DEFAULT",
   "versionGuid": "{BD3F4817-9A00-41AC-B0CC-58F78DBAE0A1}"
  },
  {
   "versionName": "UNADMIN.ProjectA",
   "versionGuid": "{F93DB9FD-6F39-45D9-A6C7-D43E69EB3076}"
  }
 ],
 "success": true
}
```

root/versions/<versionName>

resource response



JSON Response Syntax

```
{
  "versionName": "<owner.versionName>",
  "versionGuid": <guid>,
  "description": "<description>",
  "creationDate": <dateTime>,
  "modifiedDate": <dateTime>,
  "reconcileDate": <dateTime>,
  "commonAncestorDate": <dateTime>,
  "evaluationDate": <dateTime>,
  "isBeingEdited": <true | false>,
  "isBeingRead": <true | false>,
  "hasConflicts": <true | false>,
  "hasUninspectedConflicts": <true | false>,
  "isLocked": <true | false>,
  "lockOwner": "<lockOwner>",
  "lockDate": <dateTime>,
  "access": "private" | "public" | "protected",
}
```

Create version

example

create

POST only

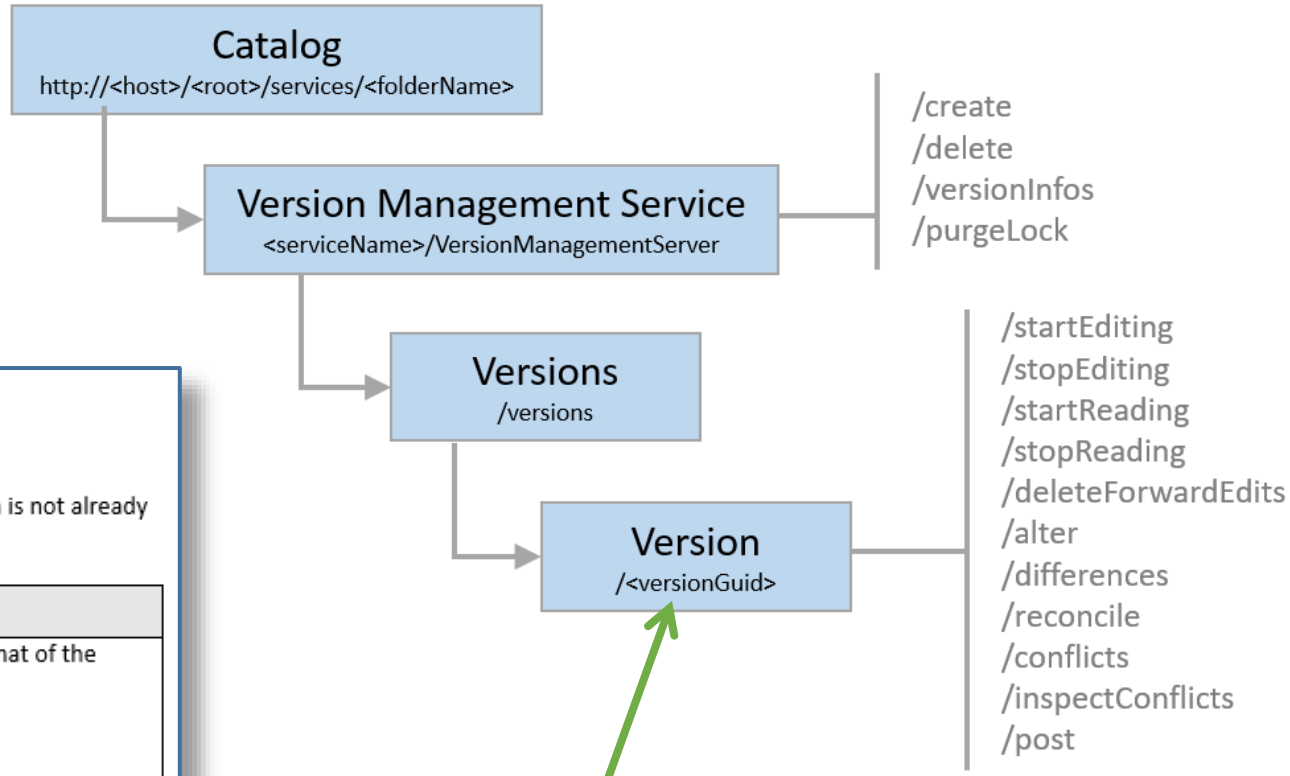
Create the named version off of DEFAULT. The version is associated with the specified feature service. During creation, the description and access (default is public) may be optionally set.

Parameter	Details
f	Description: Optional parameter to specify the output format of the response. The default response format is JSON. Values: json
versionName	Description: The name of the new version. Syntax: versionName = "version name"
description	Description: Optional parameter to specify the description of the new version. Syntax: description = "description"
accessPermission	Description: Optional parameter to specify the access permissions of the new version. The default access permission is private. Values: private | public | protected | hidden

```
POST /server/rest/services/<service-name>/VersionManagementServer/create?
token=<token>&f=json&versionName=MyNewVersion&accessPermission=private
```

Start editing

example



startEditing

POST only

Using the specified session id, start a service edit session on the version if the version is not already being edited by another user/session.

Parameter	Details
f	Description: Optional parameter to specify the output format of the response. The default response format is JSON. Values: json
sessionId	Description: The client generated session id (guid) for the version that a new service edit session is being started on; a required parameter. The session id is often passed as a logical lock token for the version being edited. Syntax : sessionId = {3F2504E0-4F89-41D3-9A0C-0305E82C3301}

- Version guid
- Session id

```
POST /server/rest/services/<service-name>/VersionManagementServer/
versions/787AD428-8C9D-4807-B97F-1FA0D265011D/startEditing?
token=<token>&f=json&sessionId={747E6DE5-A380-40FA-DC10-AAF3EF2A46AD}
```

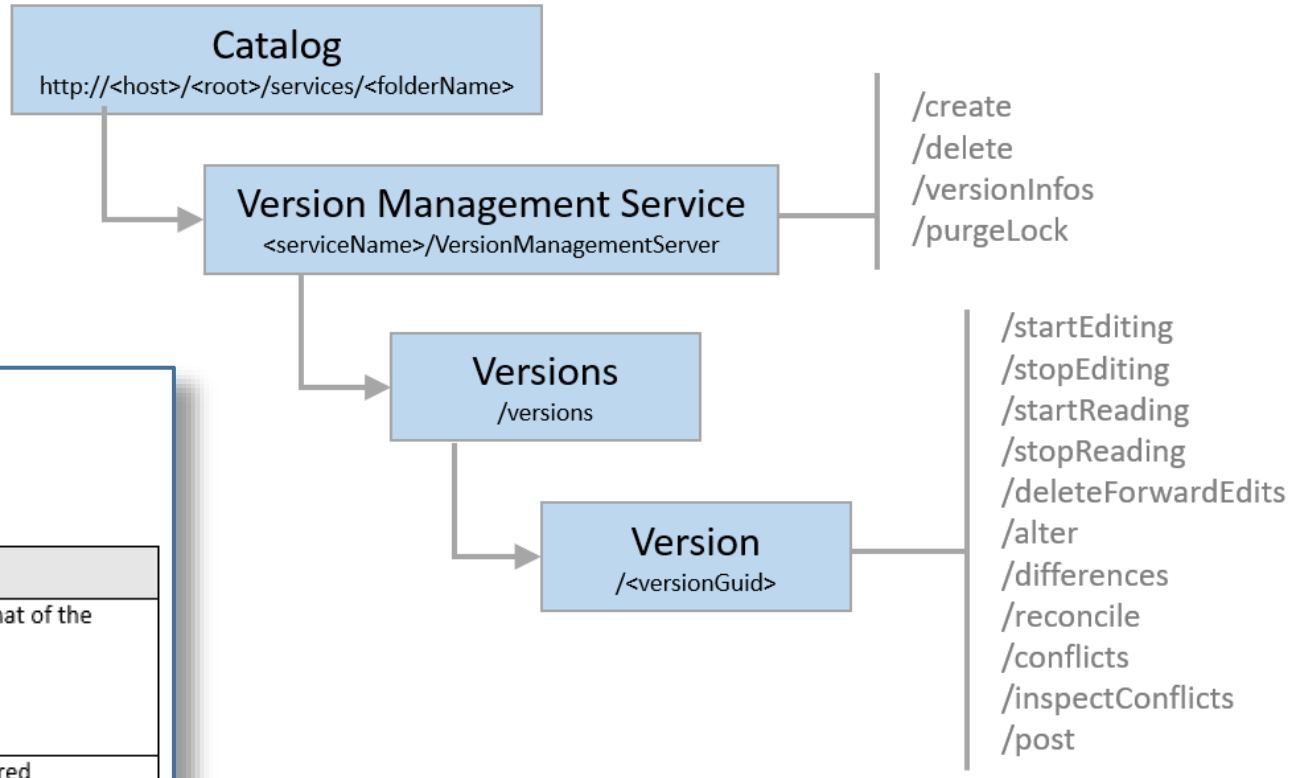
Apply edits

feature server

```
POST /server/rest/services/<service-name>/FeatureServer/applyEdits?token=<token>&
   f=json&
   rollbackOnFailure=true&
   sessionId={107E6DE5-3EE7-40FA-B855-AAF3EF2A46AD}&
   useObjectIdsForAdds=true&
   useGlobalIds=true&
   trueCurveClient=true&
   gdbVersion=MyVersion&
   honorSequenceOfEdits=true&returnEditMoment=true&
   returnServiceEditsOption=originalAndCurrentFeatures&
   usePreviousEditMoment=false&
   edits=[
     {"id":3,
      "adds":[
        {"attributes":
           {"OBJECTID":318727,
            "ASSETGROUP":2,
            "ASSETTYPE":1,
            . . .
            "NETWORKROUTENAME":"Unknown",
            "VALIDATIONSTATUS":6
           },
         "geometry":<geometry>
        }
      ]
     }
   ]
```

Stop editing

discard edits



stopEditing

POST only

Using the specified session id, stop a service edit session and save (or discard) edits.

Parameter	Details
f	Description: Optional parameter to specify the output format of the response. The default response format is JSON. Values: json
sessionId	Description: The client generated session id (guid) ; a required parameter. Syntax : sessionId = {3F2504E0-4F89-41D3-9A0C-0305E82C3301}
saveEdits	Description: Should the edits in the version be saved; this is a required parameter.

- Version guid
- Session id

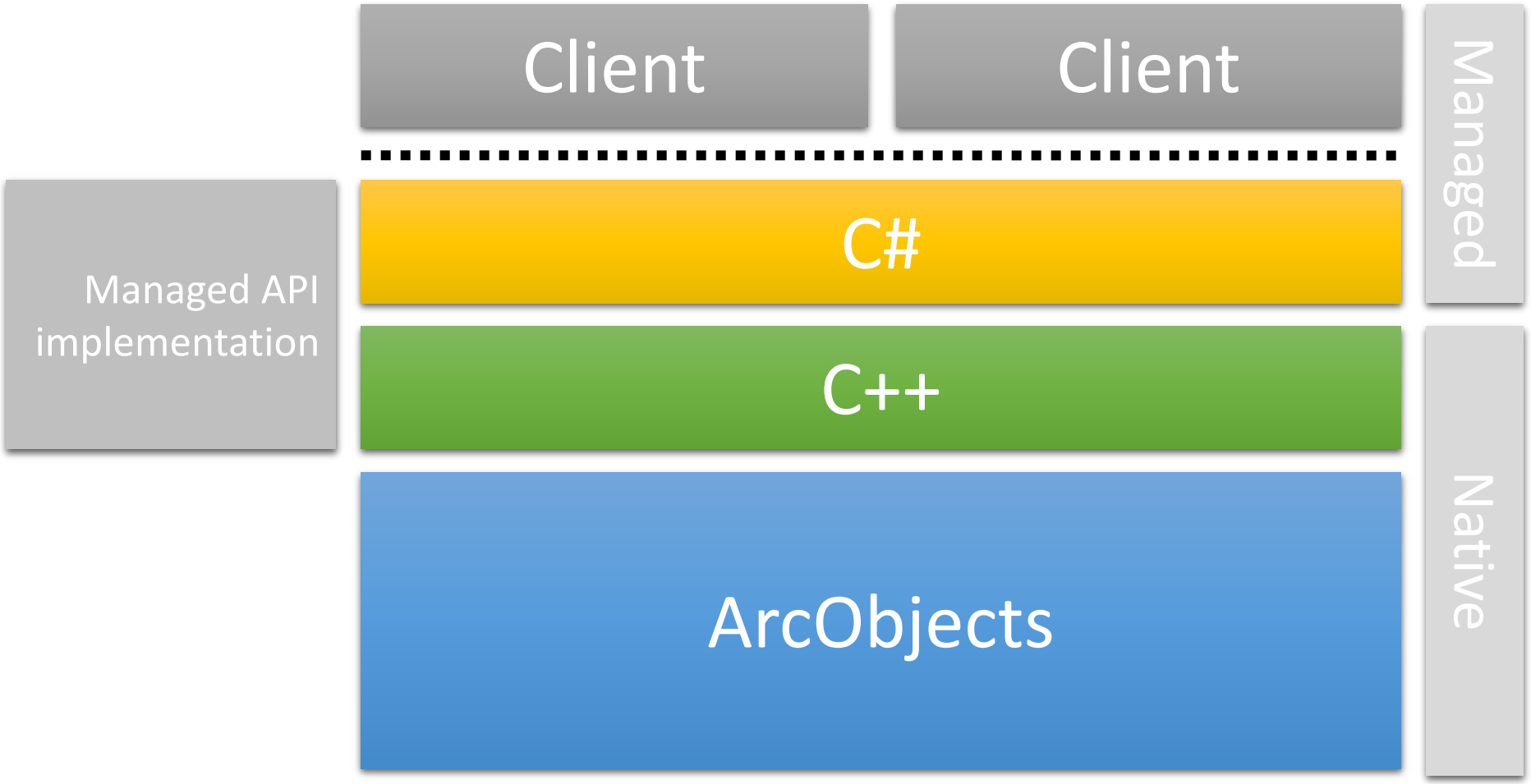
```
POST /server/rest/services/<service-name>/VersionManagementServer/
versions/787AD428-8C9D-4807-B97F-1FA0D265011D/stopEditing?
token=<token>&f=json&sessionId={747E6DE5-A380-40FA-DC10-AAF3EF2A46AD}&
saveEdits=false
```

Pro managed SDK

Erik Hoel

Core.Data

a geodatabase C# API powered by ArcObjects



API characteristics

- Designed and implemented using industry standard C# patterns and idioms
- It is minimally complete – exposes essential abstractions in only a few namespaces:
  - ArcGIS.Core.Data and sub namespaces, e.g., ArcGIS.Core.Data.Raster
- Provides a strongly typed object model
  - Minimizes object type casting
  - Helps make the API discoverable and easy to use correctly
- Offers good support for exception handling
  - Pro SDK: `ArcGIS.Core.Data.GeodatabaseGeneralException: Cannot acquire a lock.`
  - ArcObjects: `System.Runtime.InteropServices.COMException: Exception from HRESULT: 0x8004022D`

Data manipulation language

- Core.Data is a DML-only (Data Manipulation Language) API*
  - Cannot perform schema creation or modification operations:
    - Creating tables or adding fields
    - Creating domains or adding coded values
- Schema operations are performed using the GP (Geoprocessing) tools
- GP tools can be called from C# using the Geoprocessing API (ArcGIS.Desktop.Core.Geoprocessing namespace)

```
IReadOnlyList<string> args = Geoprocessing.MakeValueArray(@"C:\Data.gdb", "Places");
IGPResult gpResult         = await Geoprocessing.ExecuteToolAsync("CreateFeatureclass_management", args);

if (gpResult.IsFailed)
{
  // ...
}
```

Data definition language

new at 2.7, more at 2.8

Support users / business partners who want to create and work with their own data

- Creating file geodatabases, feature datasets, tables, and feature classes for storing results
- Creating and deleting domains, scratch datasets
- Modifying and renaming datasets
- Manipulating their own datasets (e.g., a third party application's system tables)

Geoprocessing and python will remain the preferred way to do large scale operations

Use the SchemaBuilder and the Description objects

```
// Create a SchemaBuilder object
SchemaBuilder schemaBuilder = new SchemaBuilder(geodatabase);
// Create and define a TableDefinition object
. . .
// Add the creation of the table to the list of DDL tasks
schemaBuilder.Create(tableDescription);
// Execute the DDL operation
bool success = schemaBuilder.Build();
```

Threading

Almost all of the methods in Core.Data API should be called on the Main CIM Thread (MCT)

- API reference documentation on the methods that need to run on the MCT are specified
- These methods calls should be wrapped inside the QueuedTask.Run call
- Failure to do so will result in CalledOnWrongThreadException or ConstructedOnWrongThreadException being thrown

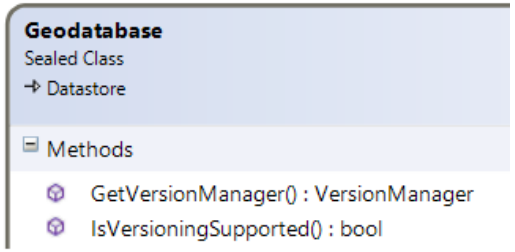
```
await QueuedTask.Run(() =>
{
  using (Geodatabase geodatabase = new Geodatabase(new DatabaseConnectionFile(new Uri(sdeFilePath))))
  {
    workspaceConnectionString = geodatabase.GetConnectionString();

    // ...
  }
});
```

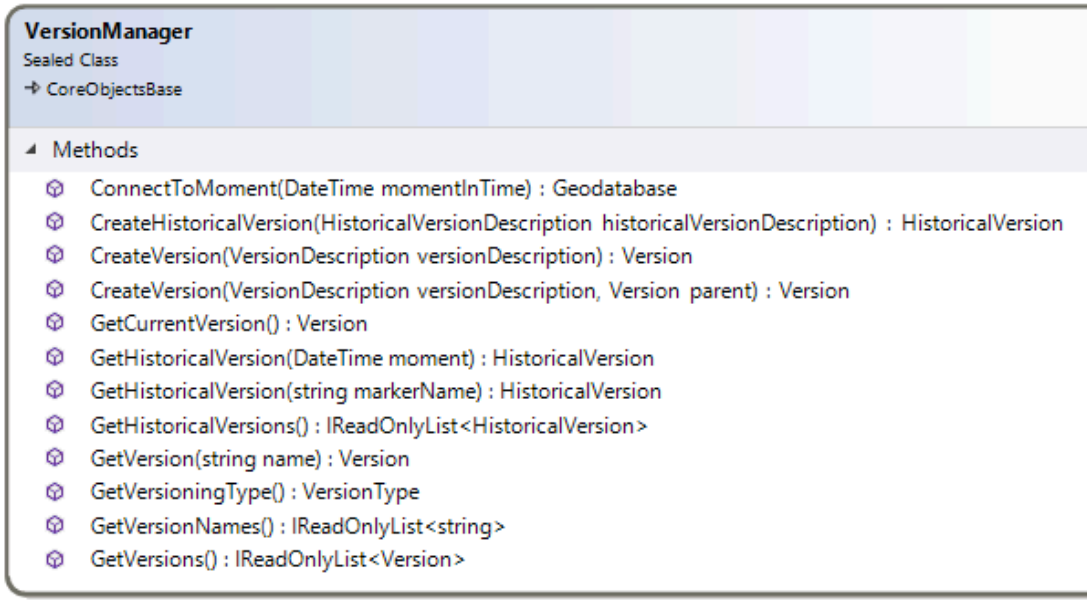
Versioning

The API provides capabilities to work with versioning (both branch and traditional versioning)

- Lists all the versions in a geodatabase, including properties of the versions
- Connect to a specific version
- Create a new version and delete an existing version
- Lists the differences between tables and feature classes from different versions
- Reconcile and post

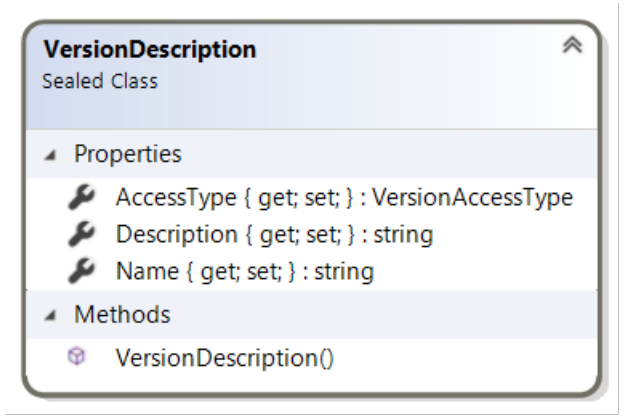
Managed SDK





VersionManager

- Available only if true is returned by `Geodatabase.IsVersioningSupported()`
- Obtained with `Geodatabase.GetVersionManager()`



The API works with both branch and traditional versioning; current capabilities include:

- Create versions
- Connect to a specific named version
- Connect to a moment
- List all versions in a geodatabase, including properties
- List differences between tables and feature classes from different versions
  - `Table.Differences()`
- Reconcile and post

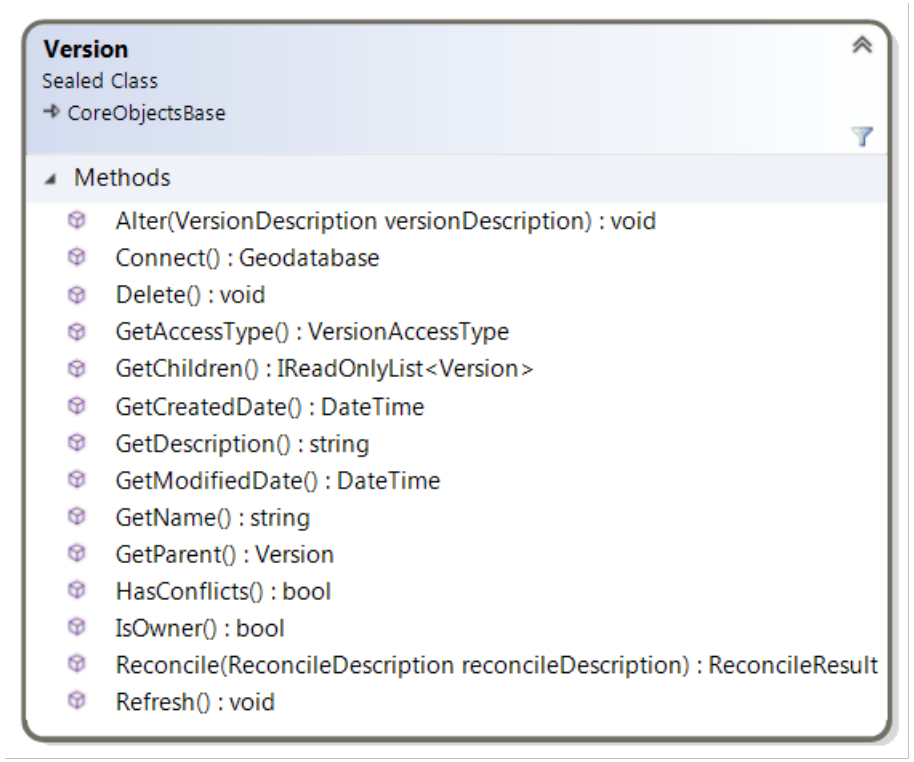
Managed SDK

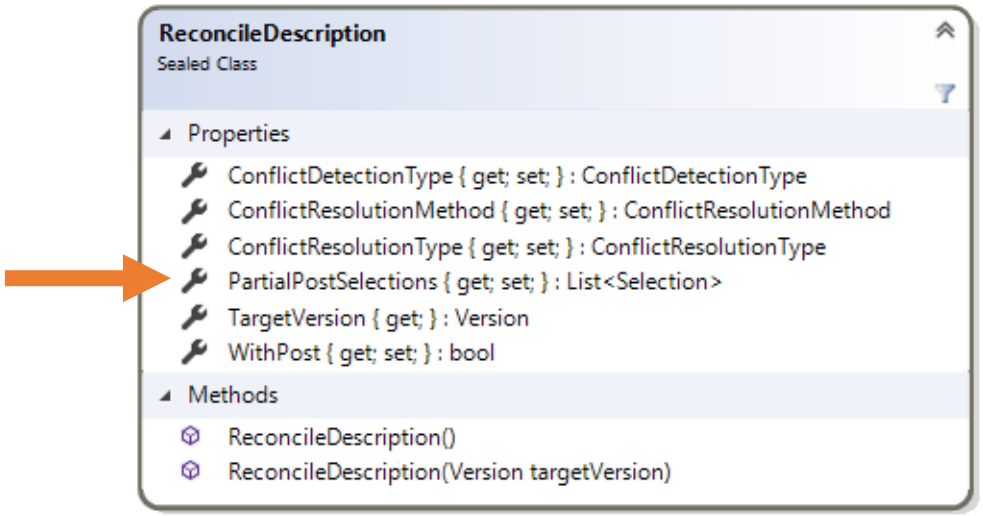
Version

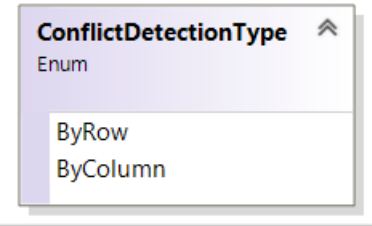
- Information about a version
- Alter or delete, reconcile and post, refresh

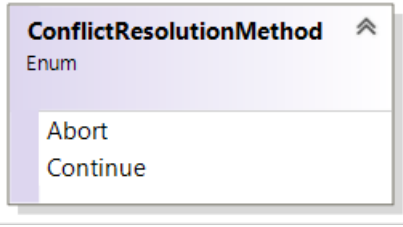
ReconcileDescription

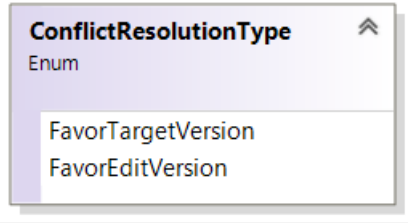
- Reconcile and post
- Detection type, resolution type/method
- New: partial posting

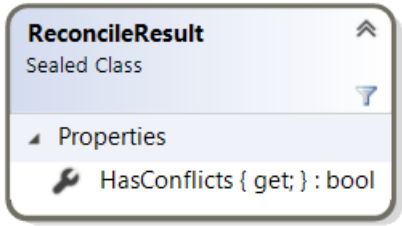












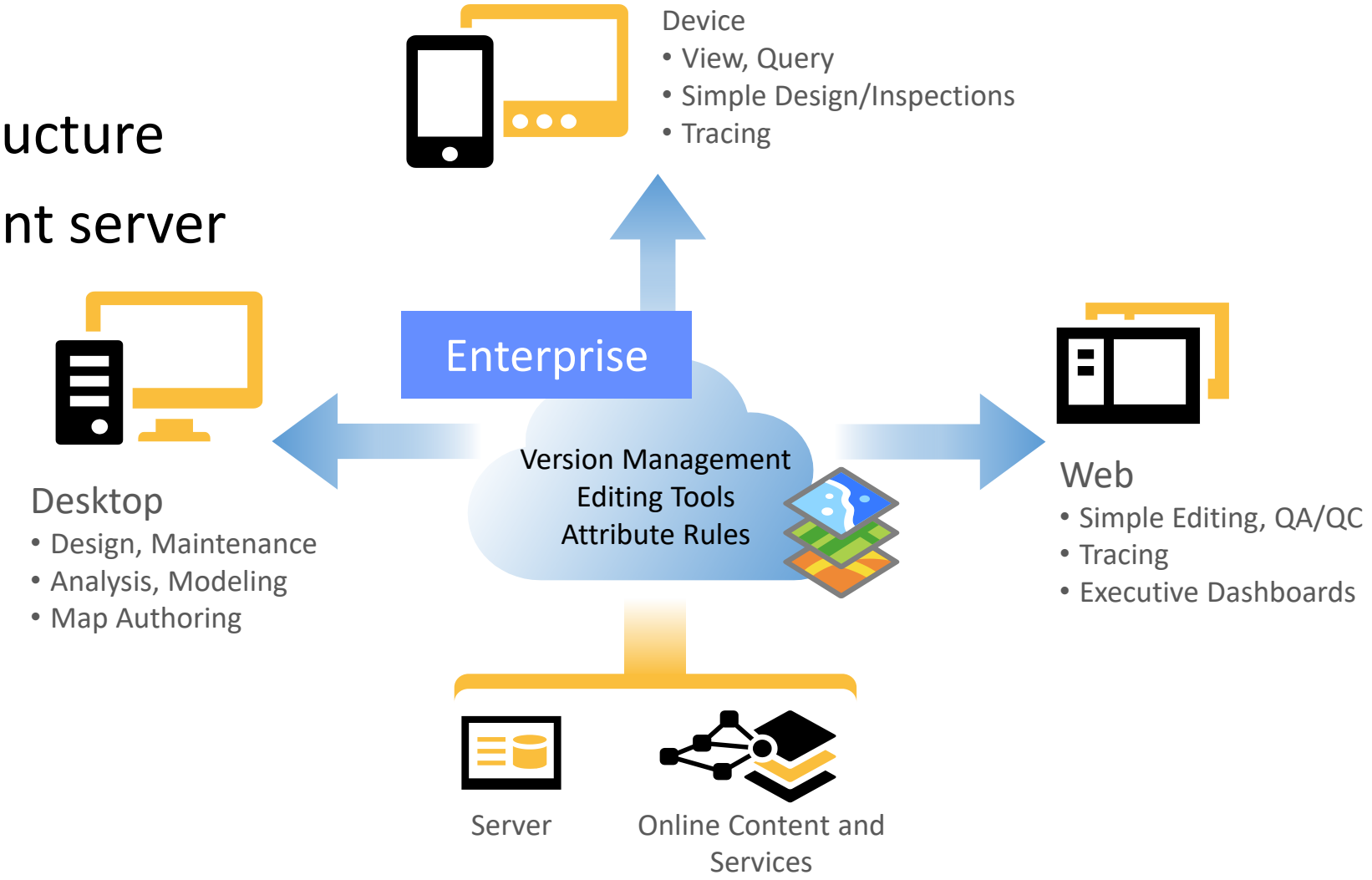
Example code

create a version, reconcile and post, and delete a version

```
const string URL = "https://sample.esri.com/server/rest/services/SampleDataset/FeatureServer";

using (Geodatabase geodatabase = new Geodatabase(new ServiceConnectionProperties(new Uri(URL))))
using (VersionManager versionManager = geodatabase.GetVersionManager())
{
  // Create a new version off the default version and connect to it.

  using (Version defaultVersion = versionManager.GetCurrentVersion())
  using (Version editVersion = versionManager.CreateVersion(new VersionDescription("CA", "", VersionAccessType.Public), defaultVersion))
  using (Geodatabase editGeodatabase = editVersion.Connect())
  {
    using (FeatureClass city = editGeodatabase.OpenDataset<FeatureClass>("0"))
    {
      Console.WriteLine("Number of features => " + city.GetCount());

      // Perform some work on the feature class.
    }

    // Reconcile and post the edit version with the default version.

    ReconcileDescription reconcileDescription = new ReconcileDescription(defaultVersion)
    {
      WithPost = true
    };

    ReconcileResult reconcileResult = editVersion.Reconcile(reconcileDescription);

    // Work is done -- delete the version.

    editVersion.Delete();
  }
}
```

Summary

- Overview
- Table and query structure
- Version management server
  - REST API
- Pro managed SDK



References

To Branch or Not to Branch (blog)

https://tinyurl.com/toBranch

Setting up data (blog)

https://tinyurl.com/settingUpData

Version administrator for branch versioned data (blog)

https://tinyurl.com/versionAdmin

Version management API (ref doc)

https://tinyurl.com/versionManagementAPI

Future enhancements

Performance and scalability

Version posting queue

Difference filtering based on a list of feature layers

Historical markers

Simplify UX for registering data as branch versioned

Please provide your feedback for this session by clicking on the session survey link directly below the video.