# CPS-506: List all known modules and revision



**CPS-506** - Getting issue details... STATUS

## Background

List all know modules and revision know by CPS. i.e list the module and revisions of yang resources in the yang_resources db table. Currently within this yang_resources table, we are storing fileName, content and checksum, we do not store module metadata. To retrieve module name and revision, we need to store these information when a yang resource is being added to the db table.

## Proposal

For the following open question on the Jira.

1. RestConf module information contain both Name and Namespace (as well as revision). Do we need both of these for uniqueness? https://datatracker.ietf.org/doc/html/rfc6020#section-5.1 states *"The names of all standard modules and submodules MUST be unique"*. The pitfall could be the word 'standard' here and it might be good to include dataspace as well. https://datatracker.ietf.org/doc/html/rfc6020#section-5.3 states that *"Name spaces for private modules are assigned by the organization owning the module without a central registry. Namespace URIs MUST be chosen so they cannot collide with standard or other enterprise namespaces, for example by using the enterprise or organization name in the namespace."*

When importing a module, according to https://datatracker.ietf.org/doc/html/rfc6020#section-5.1.1 name and revision are used to identify the module that is to be imported. Similarly, we can store name and revision of the module in our database and this will uniquely identify the module. For this, we need to add fields for module metadata in the yang-resource table. The fields would be moduleName and revision, where moduleName is the name of the module and revision is the revision of the module. This change will be going in the following file.

- cps-ri/src/main/java/org/onap/cps/spi/entities/YangResourceEntity.java
  - **YangResourceEntity.java changes**

```
@NotNull
@Column
private String moduleName;

@NotNull
@Column
private String revision;
```

Following on from this, we need to now update how we store name, revision, content in the yang-resource table.

When a request comes in to create schema set. The AdminRestController calls the cpsModuleService.createSchemaSet and passes the schemaSetName, dataspaceName and yangResourcesNameToContentMap. This createSchemaSet method validates the yang resources and then calls cpsModulePersistenceService.storeSchemaSet and passes the schemaSetName, dataspaceName and yangResourcesNameToContentMap. This method is within the class CpsModulePersistenceServiceImpl. The following changes will need to be made within this class to support the adding of moduleName and revision when storing the yang resource.

**Changes needed to be made to CpsModulePersistenceServiceImpl**

```
// need to add the following pattern in line 79
private static final Pattern RFC6020_RECOMMENDED_FILENAME_PATTERN = Pattern.compile("([\\w-]+)@(\\d{4}-\\d{2}-
\\d{2})(?:\\.yang)?", Pattern.CASE_INSENSITIVE);
// in the method synchronizeYangResources the following changes need to be added
// after line 123
final Map<String,String> metaDataMap = getModuleNameAndRevision(entry.getKey(), entry.getValue());
// after line 127
yangResourceEntity.setModuleName(metaDataMap.get("moduleName"));
yangResourceEntity.setRevision(metaDataMap.get("revision"));
// at the end of the file add the following methods
    private static HashMap<String, String> getModuleNameAndRevision(final String sourceName, final String
source) {
        final HashMap<String, String> metaDataMap = new HashMap<>();
        final var revisionSourceIdentifier =
                createIdentifierFromSourceName(checkNotNull(sourceName));

        YangTextSchemaSource tempYangTextSchemaSource = new YangTextSchemaSource(revisionSourceIdentifier) {
            @Override
            protected MoreObjects.ToStringHelper addToStringAttributes(
                    final MoreObjects.ToStringHelper toStringHelper) {
                return toStringHelper;
            }

            @Override
            public InputStream openStream() {
                return new ByteArrayInputStream(source.getBytes(StandardCharsets.UTF_8));
            }
        };
        try {
            final var dependencyInfo = YangModelDependencyInfo.forYangText(tempYangTextSchemaSource);
            metaDataMap.put("moduleName", dependencyInfo.getName());
            metaDataMap.put("revision", String.valueOf(dependencyInfo.getRevision()));
        } catch (IOException e) {
            e.printStackTrace();
        } catch (YangSyntaxErrorException e) {
            e.printStackTrace();
        }
        return metaDataMap;
    }

    private static RevisionSourceIdentifier createIdentifierFromSourceName(final String sourceName) {
        final var matcher = RFC6020_RECOMMENDED_FILENAME_PATTERN.matcher(sourceName);
        if (matcher.matches()) {
            return RevisionSourceIdentifier.create(matcher.group(1), Revision.of(matcher.group(2)));
        }
        return RevisionSourceIdentifier.create(sourceName);
    }
```

Following on from this, we need to add a liquibase step to update the yang_resources table to include the two new columns. To do this, create a new yaml file  10-add-column-to-yang-resources-table.yaml

**Liquibase Update**

```
databaseChangeLog:
  - changeset:
      id: 10
      label: addModuleNameAndRevisionColumn
      author: cps
      changes:
        - addColumn:
            tableName: yang_resource
            columns:
              - column:
                  name: module_name
                  type: TEXT
              - column:
                  name: revision
                  type: TEXT
```

The cps-ri/src/main/resources/changelog/changelog-master.yaml will also include the following at the end of the file

- `- include:`
  `file: changelog/db/changes/10-add-column-to-yang-resources-table.yaml`

With these changes, when a yang resource is being inserted into the database the module name and revision will also be included.

Finally, we need to add a method to cps-service/src/main/java/org/onap/cps/api/CpsModuleService.java to retrieve all modules and version known by CPS.

**Method added to CpsModuleService.java**

```
    /**
     * Retrieve all modules and revisions known by CPS
     * @return a list of ModuleReferences
     */
    List<ModuleReferences> getAllYangResources();
```

For this we will use an existing model called ModuleReference in the cps-service/src/main/java/org/onap/cps/spi/model/ModuleReference.java

Another method getAllYangResources will also need to be added in the cps-service/src/main/java/org/onap/cps/spi/CpsModulePersistenceService.java

**Method added to CpsModulePersistenceService.java**

```
    /**
     * Returns all YANG resources
     *
     * @return List of ModuleReferences
     */
    List<ModuleReferences> getAllYangResources();
```