TITLE Program & Catalogs APIs – SOLR Searching	NUMBER	REVISION 1.0

DIRECTV Proprietary

Prepared By

Marcos Serrano

Software Developer

Approvals

Prepared By: BSE	Date	Approved By: BSE	Date
Approved By: BSO	Date	Approved By: BSO	Date

Document Change History

Version	Issue Date	Updated By	Description
1.0			• Initial version

Table of Contents

1 Overview

1.1 *Project Overview*

The ProgramTMS application is a system used to manage program & catalogs such as Genre, Cast, Crew, Rating, Qualifier & Program Type. In order to support new a Program, Variants & Versions, the application will provide services to achieve these requirements. In addition to this, Solr application will be configure to provide text searching services to improve request-response model performance.

1.2 *Purpose*

This document defines the detailed design for Program & Catalogs APIs, as well, Solr Searching services for ProgramTMS project. This document will describe the systems, components, interactions, and classes involved in these services.

2 Solutions Design

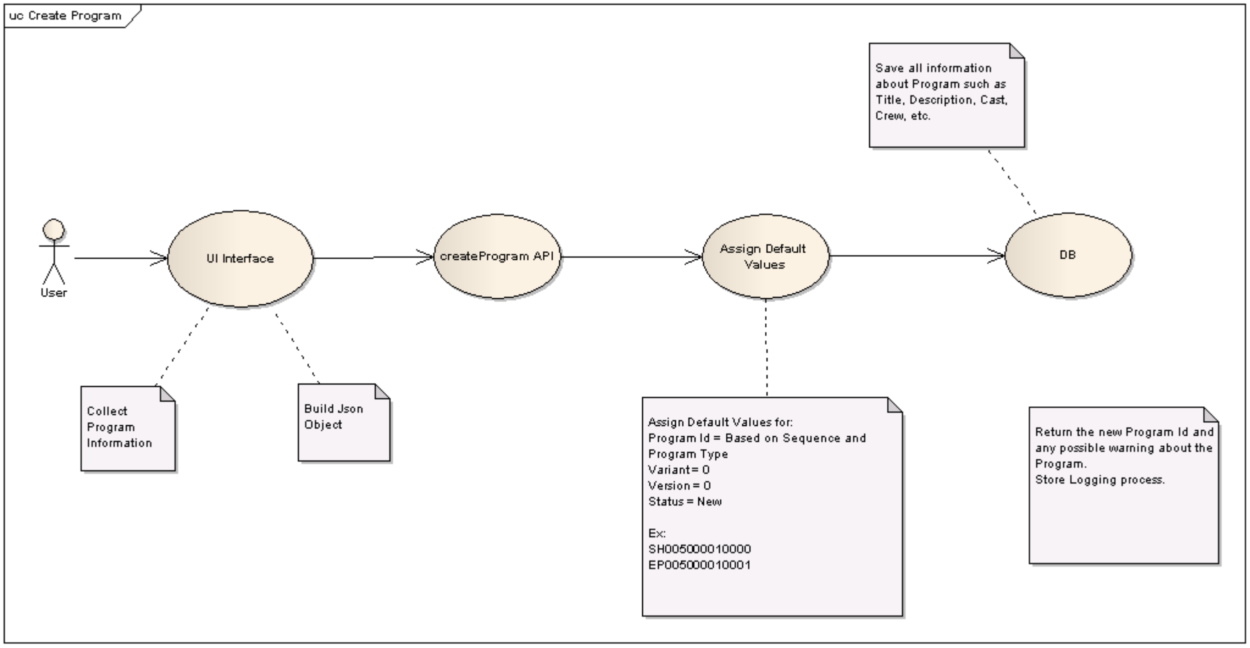
2.1 *Design Overview*

In order to support the functionality requirements for the ProgramTMS application, a number of new services will be created. The following describes the high level designs for each service:

2.2 *Create Program*

2.2.1 Use Case Diagrams

Create Program


uc Create Program
Save all information about Program such as Title, Description, Cast, Crew, etc.
User
UI Interface
createProgram API
Assign Default Values
DB
Collect Program Information
Build Json Object
Assign Default Values for:
Program Id = Based on Sequence and Program Type
Variant= 0
Version = 0
Status = New
Ex:
SH005000010000
EP005000010001
Return the new Program Id and any possible warning about the Program.
Store Logging process.

2.2.2 Tables

Id_Map

Below is a snapshot of how the table will look like after a new program is created.

ID_MAP_REC	ROOT_ID	PROGRAM_ID	VARIANT_ID	STATUS	PROGRAM_VERSION	PGM_META_ID	OWNER	PPRFID
699380	NULL	SH005000210000	0	1	0	3382201	1	1281
699381	NULL	SH005000210000	0	1	0	3382202	1	1281

Id_Map_Log

Below is a snapshot of how the table will look like after a new program is created.

ID	ID_MAP_REC	USER_ID	MODIFY_DATE
23881	699380	50	19-FEB-12 12.45.24
23882	699381	50	19-FEB-12 12.45.25

Program_Meta

Below is a snapshot of how the table will look like after a new program is created.

PGM_META_ID	VERSION	PROGRAM_ID	LANG_ID	ROOT_ID	OWNER	TYPE	…	ADULT
3382201	0	SH005000210000	1	NULL	1	5	…	NULL
3382202	0	SH005000210000	2	NULL	1	5	…	NULL

Pgm_Title

Below is a snapshot of how the table will look like after a new program is created.

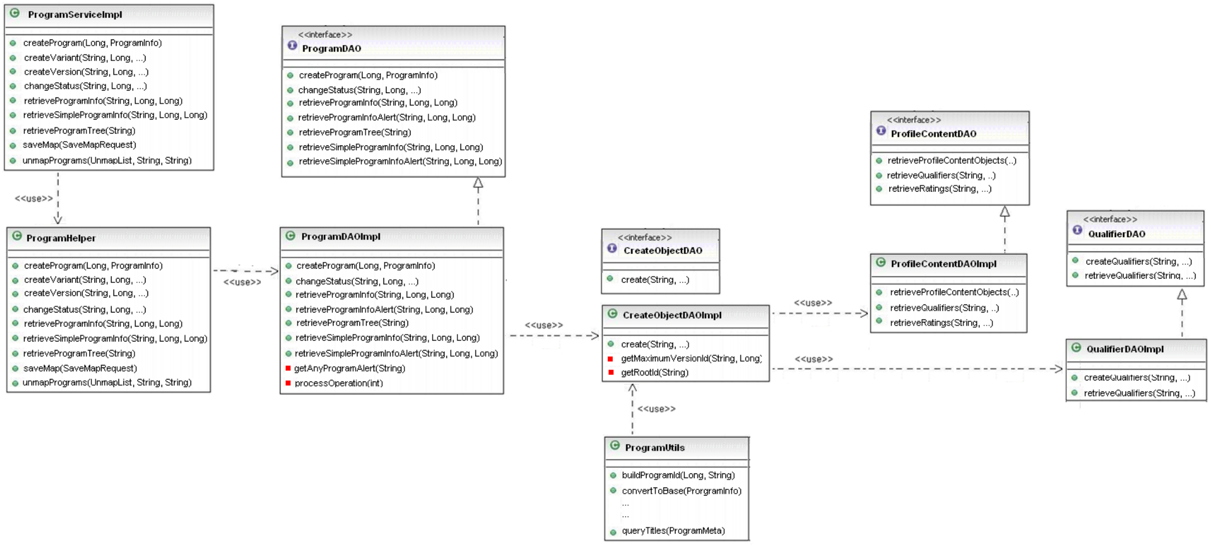
ID	LANG_ID	TEXT_SIZE	TITLE	OWNER	PGM_META_ID
5955501	1	128	TEST	1	3382201
5955502	2	128	PRUEBA	1	3382202

IMDB

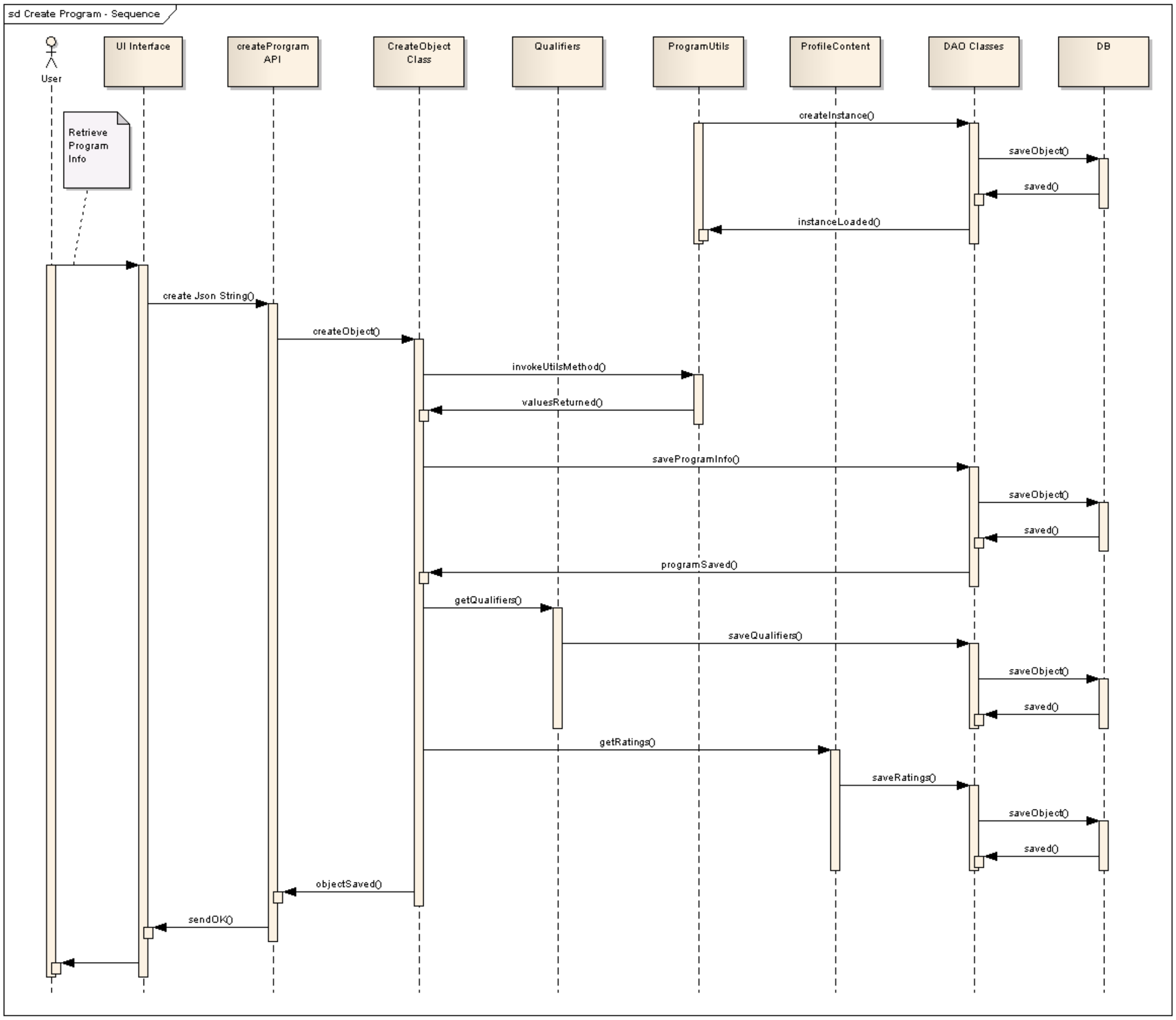
Below is a snapshot of how the table will look like after a new program is created.

ID	PROGRAM_ID	IMDB_ID	DESCRIPTION	…	IMDB_URL
21	SH005000210000	tt0111161	The Shawshank Redemption	…	3382201

2.2.3 Class Diagrams


ProgramServiceImpl
createProgram(Long, ProgramInfo)
createVariant(String, Long, ...)
createVersion(String, Long, ...)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveProgramTree(String)
saveMap(SaveMapRequest)
unmapPrograms(UnmapList, String, String)
<<use>>
ProgramHelper
createProgram(Long, ProgramInfo)
createVariant(String, Long, ...)
createVersion(String, Long, ...)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveProgramTree(String)
saveMap(SaveMapRequest)
unmapPrograms(UnmapList, String, String)
<<use>>
<<interface>>
ProgramDAO
createProgram(Long, ProgramInfo)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveProgramInfoAlert(String, Long, Long)
retrieveProgramTree(String)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveSimpleProgramInfoAlert(String, Long, Long)
ProgramDAOImpl
createProgram(Long, ProgramInfo)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveProgramInfoAlert(String, Long, Long)
retrieveProgramTree(String)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveSimpleProgramInfoAlert(String, Long, Long)
getAnyProgramAlert(String)
processOperation(int)
<<use>>
<<interface>>
CreateObjectDAO
create(String, ...)
CreateObjectDAOImpl
create(String, ...)
getMaximumVersionId(String, Long)
getRootId(String)
<<use>>
ProgramUtils
buildProgramId(Long, String)
convertToBase(ProrgramInfo)
...
...
queryTitles(ProgramMeta)
<<interface>>
ProfileContentDAO
retrieveProfileContentObjects(..)
retrieveQualifiers(String, ..)
retrieveRatings(String, ...)
<<use>>
ProfileContentDAOImpl
retrieveProfileContentObjects(..)
retrieveQualifiers(String, ..)
retrieveRatings(String, ...)
<<use>>
<<interface>>
QualifierDAO
createQualifiers(String, ...)
retrieveQualifiers(String, ...)
QualifierDAOImpl
createQualifiers(String, ...)
retrieveQualifiers(String, ...)

2.2.4 Sequence Diagrams



2.2.5 Core Components

The following core components are created:

- **ProgramServiceImpl.java** – Contains the Restful service available to use.
- **ProgramHelper.java** – Helper class.
- **ProgramDAO.java** – Interface.
- **ProgramDAOImpl.java** – Compose Program object.
- **CreateObjectDAO.java** – Interface.
- **CreateObjectDAOImpl.java** – Generic class to create the Program object.
- **ProgramUtils.java** – Class that contains useful methods.
- **ProfileContentDAO.java** – Interface.
- **ProfileContentDAOImpl.java** – Class to manage Qualifiers & Ratings for Profile Content Table.
- **QualifierDAO.java** – Interface.
- **QualifierDAOImpl.java** – Class to handle Qualifiers.

2.2.6 Test Cases

- Create Program.
  - Retrieve Program Information
  - Build Json Object
  - Post Json Object into Create Program API.
  - Compose Program Object
  - Get from Program Object any Qualifier and Save them.
  - Save Program Object
  - Log create action into ID_MAP_LOG table
  - Response Program ID and any possible warning about it.

2.2.7 Request Action

The following URL may be used to create a program:
http://*host:port*/ProgramTMS/resources/program/createProgram/**{userId}**

PARAMETER	DESCRIPTION
{userId}	Valid User Id – Numeric value.

Example:
http://*host:port*/ProgramTMS/resources/program/createProgram/**50**

HTTP Headers

The following HTTP Headers are required to complete Request process.

HTTP HEADER	VALUE
Content-Type	application/json
Accept	application/json

HTTP Method

The Request Action needs to define a HTTP **Post** method.

Request Object

The following file describes a Request Object as example:
http://*host:port*/ProgramTMS/

2.2.8 Response Action

The following file describes a Response Object as example:
http://*host:port*/ProgramTMS/

2.2.9 Create Episode - Program

In order to create an Episode, it will be required to pass the Series or Show ID into the Request object, the property identified and used for this task:

PARAMETER	DESCRIPTION
parentProgramID	Valid Program Id – Alphanumeric value.

Example:

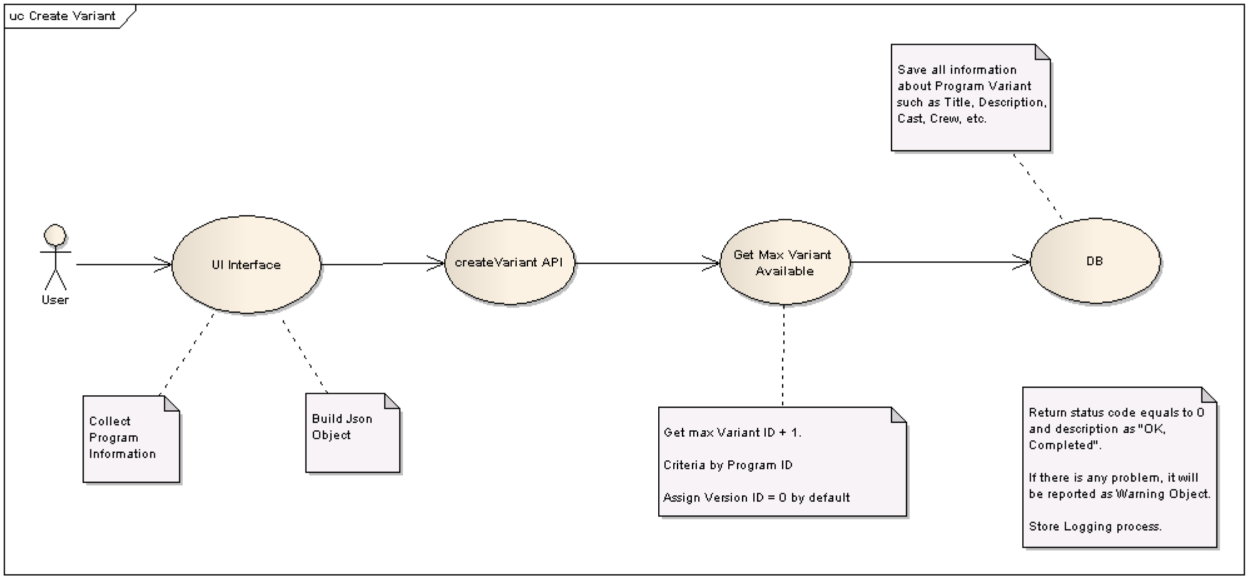
Create an Episode – brief example for Request Object.

```
"programTypeId": 1,
"programTypeDesc": "Episodes",
"programPrefixTypeId": 1,
"programPrefixTypeDesc": "EP",
"parentProgramId": "SH005000100000",
"versionId": null,
"connectorId": null,
"seasonId": null,
"seriesId": null,
"officialUrl": null,
"origAudioLang": null,
"colorCode": null,
"sourceType": null,
"animation": null,
"adult": 0,
```

2.3 *Create Variant*

2.3.1 Use Case Diagrams

Create Program Variant


uc Create Variant
Save all information about Program Variant such as Title, Description, Cast, Crew, etc.
User
UI Interface
createVariant API
Get Max Variant Available
DB
Collect Program Information
Build Json Object
Get max Variant ID + 1.
Criteria by Program ID
Assign Version ID = 0 by default
Return status code equals to 0 and description as "OK, Completed".
If there is any problem, it will be reported as Warning Object.
Store Logging process.

2.3.2 Tables

Id_Map

Below is a snapshot of how the table will look like after a new variant is created.

ID_MAP_REC	ROOT_ID	PROGRAM_ID	VARIANT_ID	STATUS	PROGRAM_VERSION	PGM_META_ID	OWNER	PPRFID
699380	NULL	SH005000210000	0	3	0	3382201	1	1281
699381	NULL	SH005000210000	0	3	0	3382202	1	1281
699402	**NULL**	**SH005000210000**	**1**	**1**	**0**	**3382225**	**1**	**1376**
699403	**NULL**	**SH005000210000**	**1**	**1**	**0**	**3382226**	**1**	**1376**

Id_Map_Log

Below is a snapshot of how the table will look like after a new variant is created.

ID	ID_MAP_REC	USER_ID	MODIFY_DATE
23899	699402	50	21-FEB-12 15.35.24
23900	699403	50	21-FEB-12 15.35.25

Program_Meta

Below is a snapshot of how the table will look like after a new variant is created.

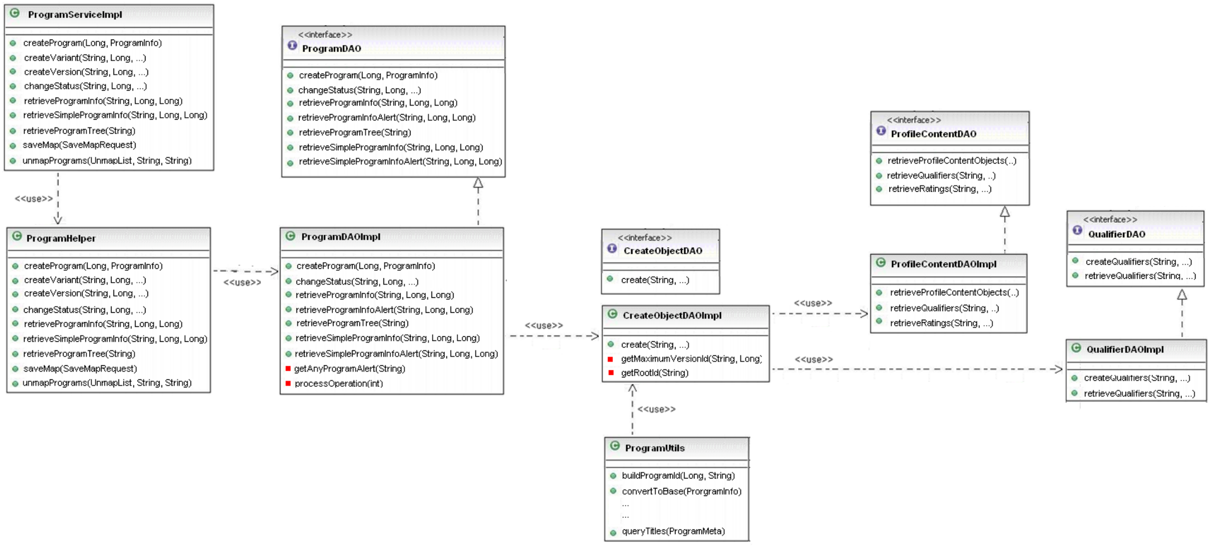
PGM_META_ID	VERSION	PROGRAM_ID	LANG_ID	ROOT_ID	OWNER	TYPE	…	ADULT
3382225	0	SH005000210000	1	NULL	1	5	…	NULL
3382226	0	SH005000210000	2	NULL	1	5	…	NULL

Pgm_Title

Below is a snapshot of how the table will look like after a new variant is created.

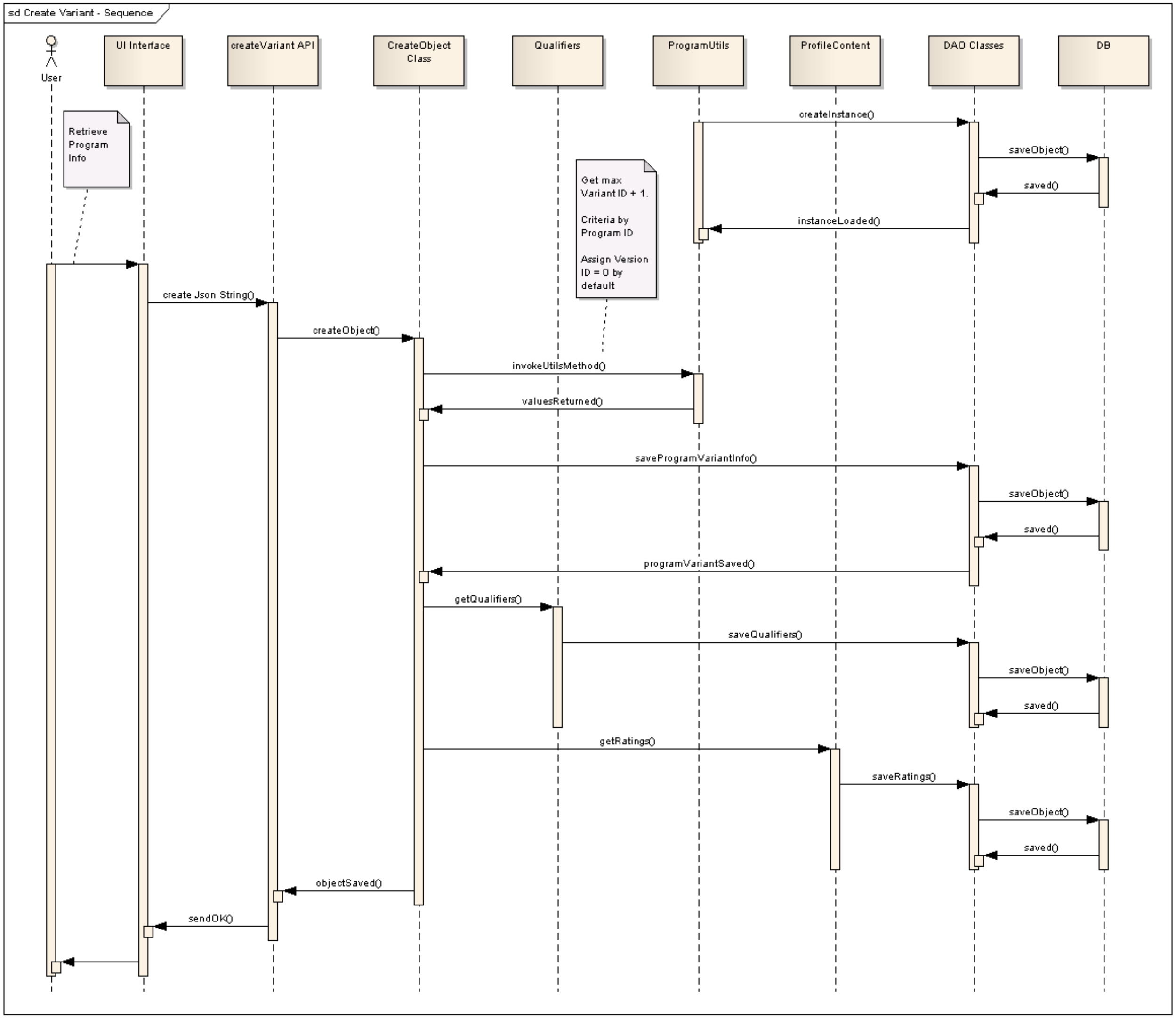
ID	LANG_ID	TEXT_SIZE	TITLE	OWNER	PGM_META_ID
5955533	1	128	TEST VARIANT 1	1	3382225
5955534	2	128	PRUEBA VARIANT 1	1	3382226

2.3.3 Class Diagrams


ProgramServiceImpl
createProgram(Long, ProgramInfo)
createVariant(String, Long, ...)
createVersion(String, Long, ...)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveProgramTree(String)
saveMap(SaveMapRequest)
unmapPrograms(UnmapList, String, String)
<<interface>>
ProgramDAO
createProgram(Long, ProgramInfo)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveProgramInfoAlert(String, Long, Long)
retrieveProgramTree(String)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveSimpleProgramInfoAlert(String, Long, Long)
<<interface>>
ProfileContentDAO
retrieveProfileContentObjects(..)
retrieveQualifiers(String, ..)
retrieveRatings(String, ...)
<<use>>
ProgramHelper
createProgram(Long, ProgramInfo)
createVariant(String, Long, ...)
createVersion(String, Long, ...)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveProgramTree(String)
saveMap(SaveMapRequest)
unmapPrograms(UnmapList, String, String)
<<use>>
ProgramDAOImpl
createProgram(Long, ProgramInfo)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveProgramInfoAlert(String, Long, Long)
retrieveProgramTree(String)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveSimpleProgramInfoAlert(String, Long, Long)
getAnyProgramAlert(String)
processOperation(int)
<<use>>
<<interface>>
CreateObjectDAO
create(String, ...)
CreateObjectDAOImpl
create(String, ...)
getMaximumVersionId(String, Long)
getRootId(String)
<<use>>
ProfileContentDAOImpl
retrieveProfileContentObjects(..)
retrieveQualifiers(String, ..)
retrieveRatings(String, ...)
<<interface>>
QualifierDAO
createQualifiers(String, ...)
retrieveQualifiers(String, ...)
<<use>>
QualifierDAOImpl
createQualifiers(String, ...)
retrieveQualifiers(String, ...)
<<use>>
ProgramUtils
buildProgramId(Long, String)
convertToBase(ProrgramInfo)
...
...
queryTitles(ProgramMeta)

2.3.4 Sequence Diagrams

Create Program Variant



2.3.5 Core Components

The following core components are created:

- **ProgramServiceImpl.java** – Contains the Restful service available to use.
- **ProgramHelper.java** – Helper class.
- **ProgramDAO.java** – Interface.
- **ProgramDAOImpl.java** – Compose Program object.
- **CreateObjectDAO.java** – Interface.
- **CreateObjectDAOImpl.java** – Generic class to create the Program object.
- **ProgramUtils.java** – Class that contains useful methods.
- **ProfileContentDAO.java** – Interface.
- **ProfileContentDAOImpl.java** – Class to manage Qualifiers & Ratings for Profile Content Table.
- **QualifierDAO.java** – Interface.
- **QualifierDAOImpl.java** – Class to handle Qualifiers.

2.3.6 Test Cases

- Create Variant.
  - Retrieve Variant Information
  - Build Json Object
  - Post Json Object into Create Variant API.
  - Compose Variant Object
  - Get from Variant Object any Qualifier and Save them.
  - Save Variant Object
  - Log create action into ID_MAP_LOG table
  - Response OK to complete process.

2.3.7 Request Action

The following URL may be used to create a variant:
http://*host:port*/ProgramTMS/resources/program/createVariant/**{programId}**&**{variantId}**&**{versionId}**&**{userId}**

PARAMETER	DESCRIPTION
{programId}	Valid Program Id – Alphanumeric value.
{variantId}	Valid Variant Id – Numeric value.
{versionId}	Valid Version Id – Numeric value.
{userId}	Valid User Id – Numeric value.

Example:
http://*host:port*/ProgramTMS/resources/program/createVariant/**SH005000010000**&**0**&**0**&**50**

HTTP Headers

The following HTTP Headers are required to complete Request process.

HTTP HEADER	VALUE
Content-Type	application/json
Accept	application/json

HTTP Method

The Request Action needs to define a HTTP **Post** method.

Request Object

The following file describes a Request Object as example:
http://*host:port*/ProgramTMS/

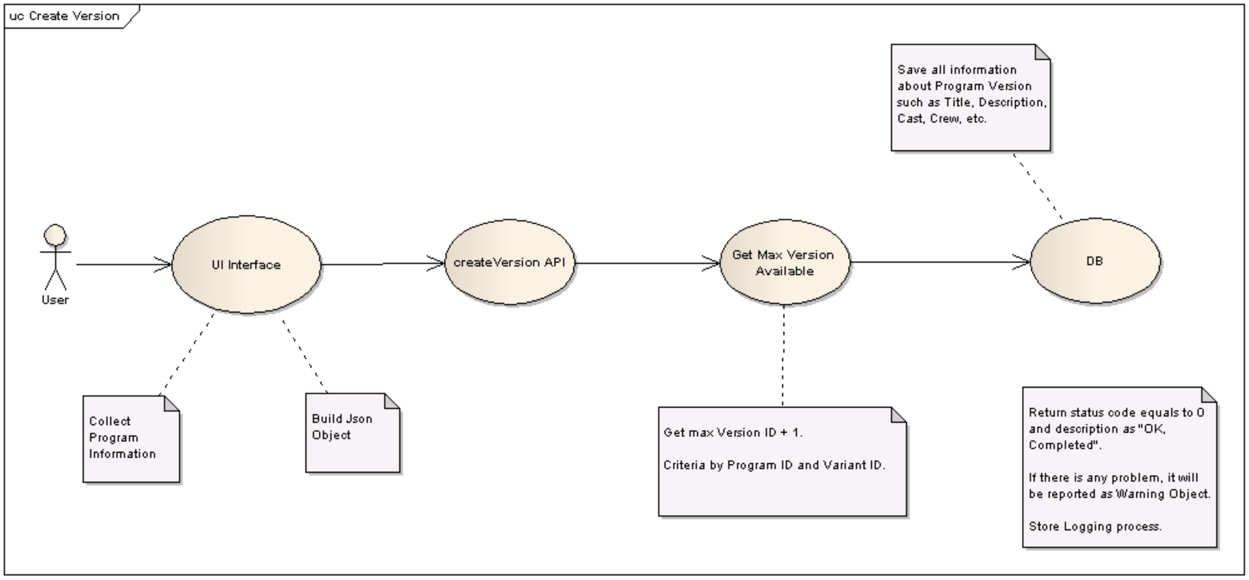
2.3.8 Response Action

The following file describes a Response Object as example:
http://*host:port*/ProgramTMS/

2.4 *Create Version*

2.4.1 Use Case Diagrams

Create Program Version


uc Create Version
User
UI Interface
createVersion API
Get Max Version Available
DB
Save all information about Program Version such as Title, Description, Cast, Crew, etc.
Collect Program Information
Build Json Object
Get max Version ID + 1.
Criteria by Program ID and Variant ID.
Return status code equals to 0 and description as "OK, Completed".
If there is any problem, it will be reported as Warning Object.
Store Logging process.

2.4.2 Tables

Id_Map

Below is a snapshot of how the table will look like after a new version is created.

ID_MAP_REC	ROOT_ID	PROGRAM_ID	VARIANT_ID	STATUS	PROGRAM_VERSION	PGM_META_ID	OWNER	PPRFID
699380	NULL	SH005000210000	0	3	0	3382201	1	1281
699381	NULL	SH005000210000	0	3	0	3382202	1	1281
699415	**NULL**	**SH005000210000**	**0**	**1**	**1**	**3382228**	**1**	**1376**
699416	**NULL**	**SH005000210000**	**0**	**1**	**1**	**3382229**	**1**	**1376**

Id_Map_Log

Below is a snapshot of how the table will look like after a new version is created.

ID	ID_MAP_REC	USER_ID	MODIFY_DATE
23915	699415	50	21-FEB-12 17.15.24
23916	699416	50	21-FEB-12 17.15.25

Program_Meta

Below is a snapshot of how the table will look like after a new version is created.

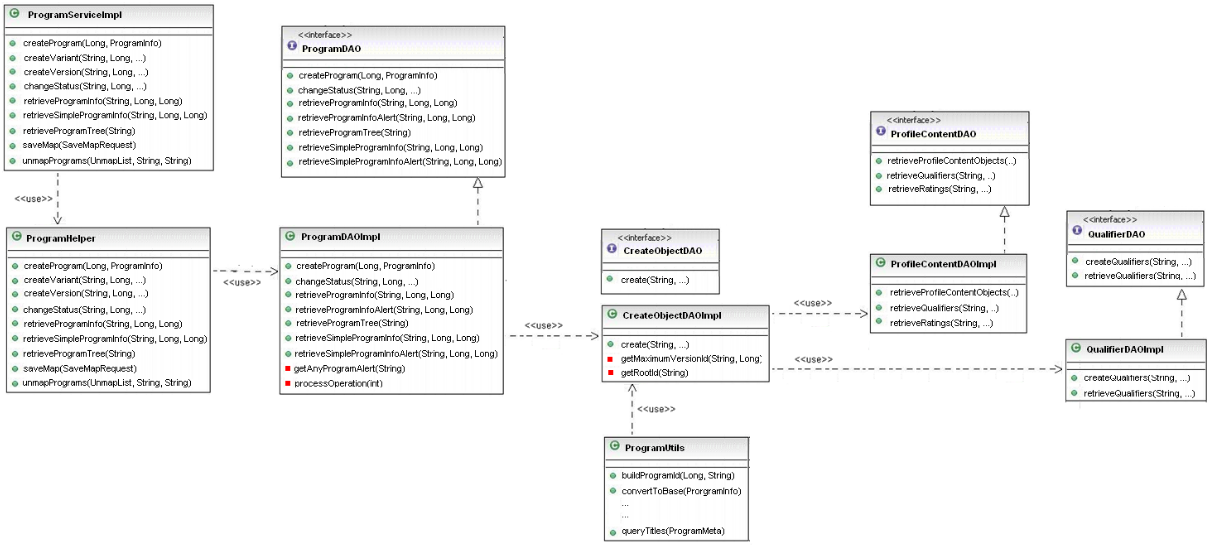
PGM_META_ID	VERSION	PROGRAM_ID	LANG_ID	ROOT_ID	OWNER	TYPE	...	ADULT
3382228	0	SH005000210000	1	NULL	1	5	...	NULL
3382229	0	SH005000210000	2	NULL	1	5	...	NULL

Pgm_Title

Below is a snapshot of how the table will look like after a new version is created.

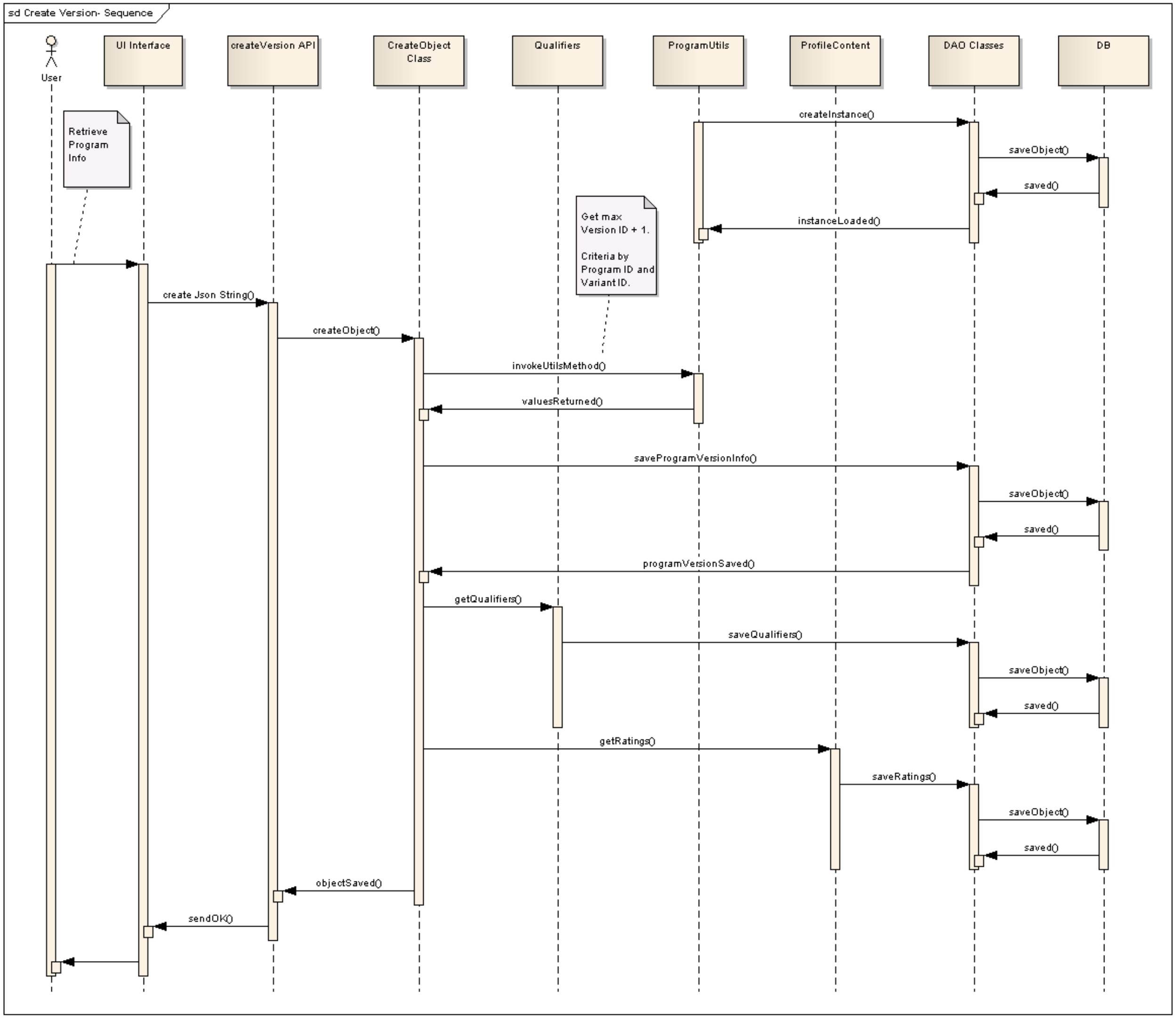
ID	LANG_ID	TEXT_SIZE	TITLE	OWNER	PGM_META_ID
5955539	1	128	TEST VERSION 1	1	3382228
5955540	2	128	PRUEBA VERSION 1	1	3382229

2.4.3 Class Diagrams


ProgramServiceImpl
createProgram(Long, ProgramInfo)
createVariant(String, Long, ...)
createVersion(String, Long, ...)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveProgramTree(String)
saveMap(SaveMapRequest)
unmapPrograms(UnmapList, String, String)
<<interface>>
ProgramDAO
createProgram(Long, ProgramInfo)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveProgramInfoAlert(String, Long, Long)
retrieveProgramTree(String)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveSimpleProgramInfoAlert(String, Long, Long)
<<use>>
ProgramHelper
createProgram(Long, ProgramInfo)
createVariant(String, Long, ...)
createVersion(String, Long, ...)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveProgramTree(String)
saveMap(SaveMapRequest)
unmapPrograms(UnmapList, String, String)
<<use>>
ProgramDAOImpl
createProgram(Long, ProgramInfo)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveProgramInfoAlert(String, Long, Long)
retrieveProgramTree(String)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveSimpleProgramInfoAlert(String, Long, Long)
getAnyProgramAlert(String)
processOperation(int)
<<use>>
<<interface>>
CreateObjectDAO
create(String, ...)
CreateObjectDAOImpl
create(String, ...)
getMaximumVersionId(String, Long)
getRootId(String)
<<use>>
ProgramUtils
buildProgramId(Long, String)
convertToBase(ProrgramInfo)
...
...
queryTitles(ProgramMeta)
<<interface>>
ProfileContentDAO
retrieveProfileContentObjects(..)
retrieveQualifiers(String, ..)
retrieveRatings(String, ...)
<<use>>
ProfileContentDAOImpl
retrieveProfileContentObjects(..)
retrieveQualifiers(String, ..)
retrieveRatings(String, ...)
<<interface>>
QualifierDAO
createQualifiers(String, ...)
retrieveQualifiers(String, ...)
<<use>>
QualifierDAOImpl
createQualifiers(String, ...)
retrieveQualifiers(String, ...)

2.4.4 Sequence Diagrams

Create Program Version



2.4.5 Core Components

The following core components are created:

- **ProgramServiceImpl.java** – Contains the Restful service available to use.
- **ProgramHelper.java** – Helper class.
- **ProgramDAO.java** – Interface.
- **ProgramDAOImpl.java** – Compose Program object.
- **CreateObjectDAO.java** – Interface.
- **CreateObjectDAOImpl.java** – Generic class to create the Program object.
- **ProgramUtils.java** – Class that contains useful methods.
- **ProfileContentDAO.java** – Interface.
- **ProfileContentDAOImpl.java** – Class to manage Qualifiers & Ratings for Profile Content Table.
- **QualifierDAO.java** – Interface.
- **QualifierDAOImpl.java** – Class to handle Qualifiers.

2.4.6 Test Cases

- Create Version.
    - Retrieve Version Information
    - Build Json Object
    - Post Json Object into Create Version API.
    - Compose Version Object
    - Get from Version Object any Qualifier and Save them.
    - Save Version Object
    - Log create action into ID_MAP_LOG table
    - Response OK to complete process.

2.4.7 Request Action

The following URL may be used to create a version:
http://*host:port*/ProgramTMS/resources/program/createVersion/**{programId}**&**{variantId}**&**{versionId}**&**{userId}**

PARAMETER	DESCRIPTION
{programId}	Valid Program Id – Alphanumeric value.
{variantId}	Valid Variant Id – Numeric value.
{versionId}	Valid Version Id – Numeric value.
{userId}	Valid User Id – Numeric value.

Example:
http://*host:port*/ProgramTMS/resources/program/createVersion/ **SH005000010000**&**0**&**0**&**50**

HTTP Headers

The following HTTP Headers are required to complete Request process.

HTTP HEADER	VALUE
Content-Type	application/json
Accept	application/json

HTTP Method

The Request Action needs to define a HTTP **Post** method.

Request Object

The following file describes a Request Object as example:
http://*host:port*/ProgramTMS/

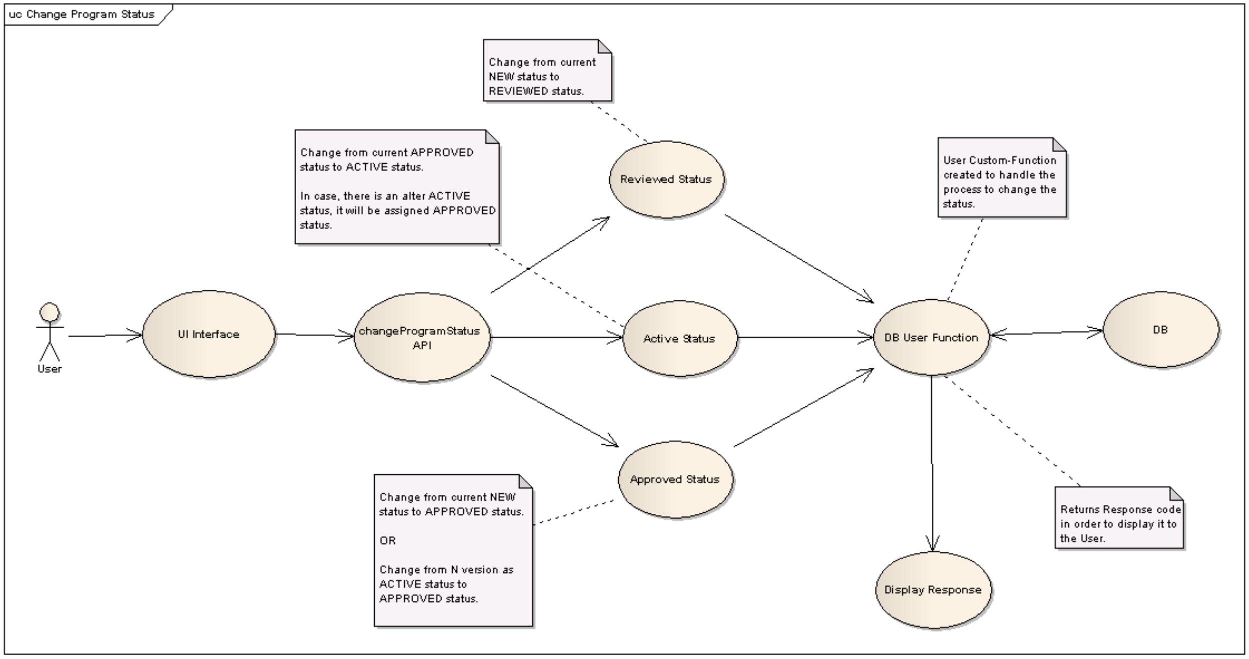
2.4.8 Response Action

The following file describes a Response Object as example:
http://*host:port*/ProgramTMS/

2.5 *Change Program Status*

2.5.1 Use Case Diagrams

Create Product / Product Summary


uc Change Program Status
Change from current NEW status to REVIEWED status.
Change from current APPROVED status to ACTIVE status.
In case, there is an alter ACTIVE status, it will be assigned APPROVED status.
User Custom-Function created to handle the process to change the status.
User
UI Interface
changeProgramStatus API
Reviewed Status
Active Status
Approved Status
DB User Function
DB
Change from current NEW status to APPROVED status.
OR
Change from N version as ACTIVE status to APPROVED status.
Returns Response code in order to display it to the User.
Display Response

2.5.2 Tables

Id_Map

Below is a snapshot of how the table will look like after a change is completed from NEW to APPROVED status.

NEW status

ID_MAP_REC	ROOT_ID	PROGRAM_ID	VARIANT_ID	STATUS	PROGRAM_VERSION	PGM_META_ID	OWNER	PPRFID
699380	NULL	SH005000210000	0	**1**	0	3382201	1	1281
699381	NULL	SH005000210000	0	**1**	0	3382202	1	1281

APPROVED status

ID_MAP_REC	ROOT_ID	PROGRAM_ID	VARIANT_ID	STATUS	PROGRAM_VERSION	PGM_META_ID	OWNER	PPRFID
699380	NULL	SH005000210000	0	**3**	0	3382201	1	1281
699381	NULL	SH005000210000	0	**3**	0	3382202	1	1281

Id_Map_Log

Below is a snapshot of how the table will look like after a change status is completed.

ID	ID_MAP_REC	USER_ID	MODIFY_DATE
23928	699380	50	23-FEB-12 10.05.24
23929	699381	50	23-FEB-12 10.05.25

2.5.3 Database User Custom Function

FN_CHANGE_PROGRAM_STATUS

The FN_CHANGE_PROGRAM_STATUS function will handle the process to change the Program status.

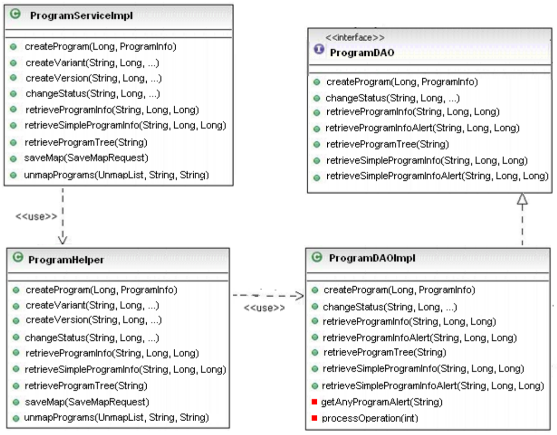
The following table describes the parameters need to pass into the DB function.

PARAMETER	TYPE	DESCRIPTION
prg_ID	ID_MAP.PROGRAM_ID%type	Valid Program ID – Alphanumeric value
variantID	ID_MAP.VARIANT_ID%type	Valid Variant ID – Numeric value
versionID	ID_MAP_PROGRAM_VERSION%type	Valid Version ID – Numeric value
userID	ID_MAP_LOG.USER_ID%type	Valid User ID – Numeric value
statusType	PGM_STATUS.ID%type	Valid Status Type – Numeric value
operationType	Integer	Either 0=Change or 1=Delete value.

The following file describes the DB function:

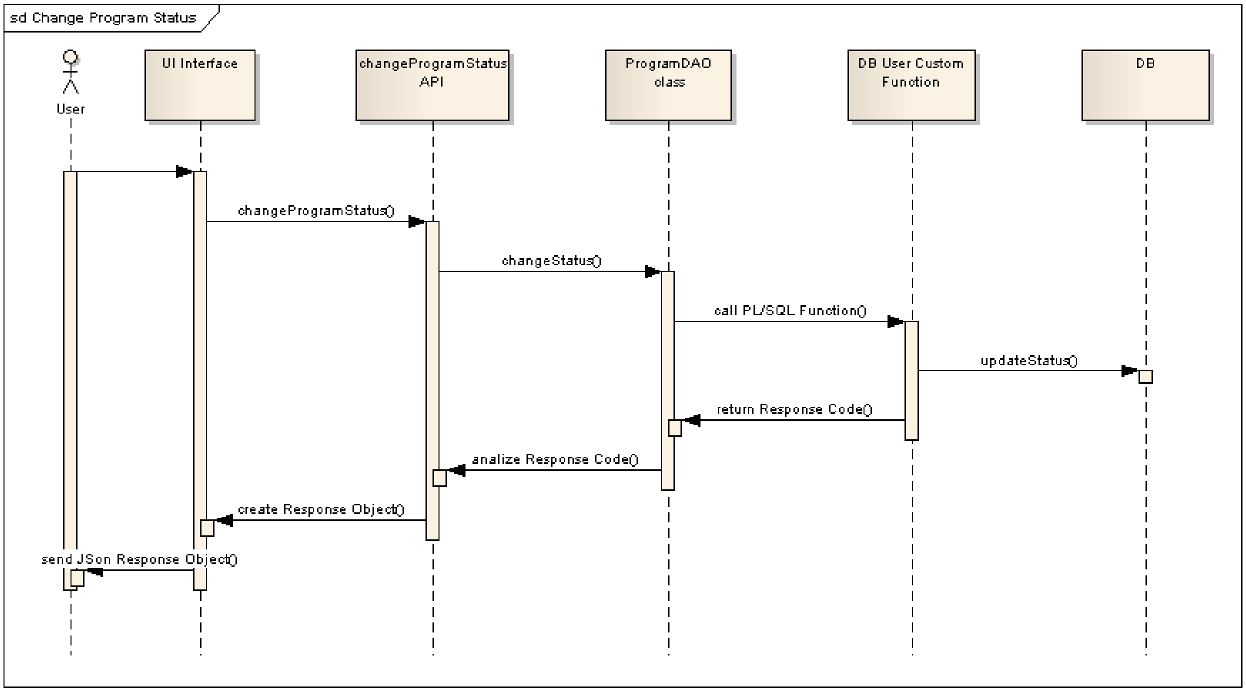
FN_CHANGE_PROGRAM_STATUS.txt

2.5.4 Class Diagrams


ProgramServiceImpl
createProgram(Long, ProgramInfo)
createVariant(String, Long, ...)
createVersion(String, Long, ...)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveProgramTree(String)
saveMap(SaveMapRequest)
unmapPrograms(UnmapList, String, String)
<<use>>
<<interface>>
ProgramDAO
createProgram(Long, ProgramInfo)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveProgramInfoAlert(String, Long, Long)
retrieveProgramTree(String)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveSimpleProgramInfoAlert(String, Long, Long)
ProgramHelper
createProgram(Long, ProgramInfo)
createVariant(String, Long, ...)
createVersion(String, Long, ...)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveProgramTree(String)
saveMap(SaveMapRequest)
unmapPrograms(UnmapList, String, String)
<<use>>
ProgramDAOImpl
createProgram(Long, ProgramInfo)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveProgramInfoAlert(String, Long, Long)
retrieveProgramTree(String)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveSimpleProgramInfoAlert(String, Long, Long)
getAnyProgramAlert(String)
processOperation(int)

2.5.5 Sequence Diagrams

Change Program Status



2.5.6 Core Components

The following core components are created:

- **ProgramServiceImpl.java** – Contains the Restful service available to use.
- **ProgramHelper.java** – Helper class.
- **ProgramDAO.java** – Interface.
- **ProgramDAOImpl.java** – Compose Program object.

2.5.7 Test Cases

- Change from NEW status to REVIEWED status.
  - Input program, variant, version & user IDs, as well, value 2 as REVIEWED status to the parameters.
  - Response OK Completed.
- Change from NEW status to APPROVED status.
  - Input program, variant, version & user IDs, as well, value 11 as APPROVED status to the parameters.
  - Response OK Completed.
- Change from APPROVED status to ACTIVE status.
  - Input program, variant, version & user IDs, as well, value 3 as ACTIVE status to the parameters.
  - In case, there is other version as ACTIVE status, it will be change it to APPROVED status automatically. There must be only one version as ACTIVE status.
  - Response OK Completed.

2.5.8 Request Action

The following URL may be used to change program status:
http://*host:port*/ProgramTMS/resources/program/changeProgramStatus/programId=**{programId}**&variantId=**{variantId}**&versionId=**{versionId}**&statusId=**{statusId}**&userId=**{userId}**

PARAMETER	DESCRIPTION
{programId}	Valid Program Id – Alphanumeric value.
{variantId}	Valid Variant Id – Numeric value.
{versionId}	Valid Version Id – Numeric value.
{statusId}	Valid Status Id – Numeric value – Options {2=Reviewed, 11=Approved, 3=Active}
{userId}	Valid User Id – Numeric value.

Example, change from NEW status to REVIEWED status:
http://*host:port*/ProgramTMS/resources/program/changeProgramStatus/programId=SH005000440000&variantId=1&versionId=1&statusId=2&userId=50

Example, change from NEW status to APPROVED status:
http://*host:port*/ProgramTMS/resources/program/changeProgramStatus/programId=SH005000440000&variantId=1&versionId=2&statusId=11&userId=50

Example, change from APPROVED status to ACTIVE status:
http://*host:port*/ProgramTMS/resources/program/changeProgramStatus/programId=SH005000440000&variantId=1&versionId=2&statusId=3&userId=50

HTTP Headers

The following HTTP Headers are required to complete Request process.

HTTP HEADER	VALUE
Content-Type	application/json
Accept	application/json

HTTP Method

The Request Action needs to define a HTTP **Get** method.

2.5.9 Response Action

The following file describes a Response Object as example:

```
{
   "statusCode": 0,
   "statusDescription": "OK completed"
}
```

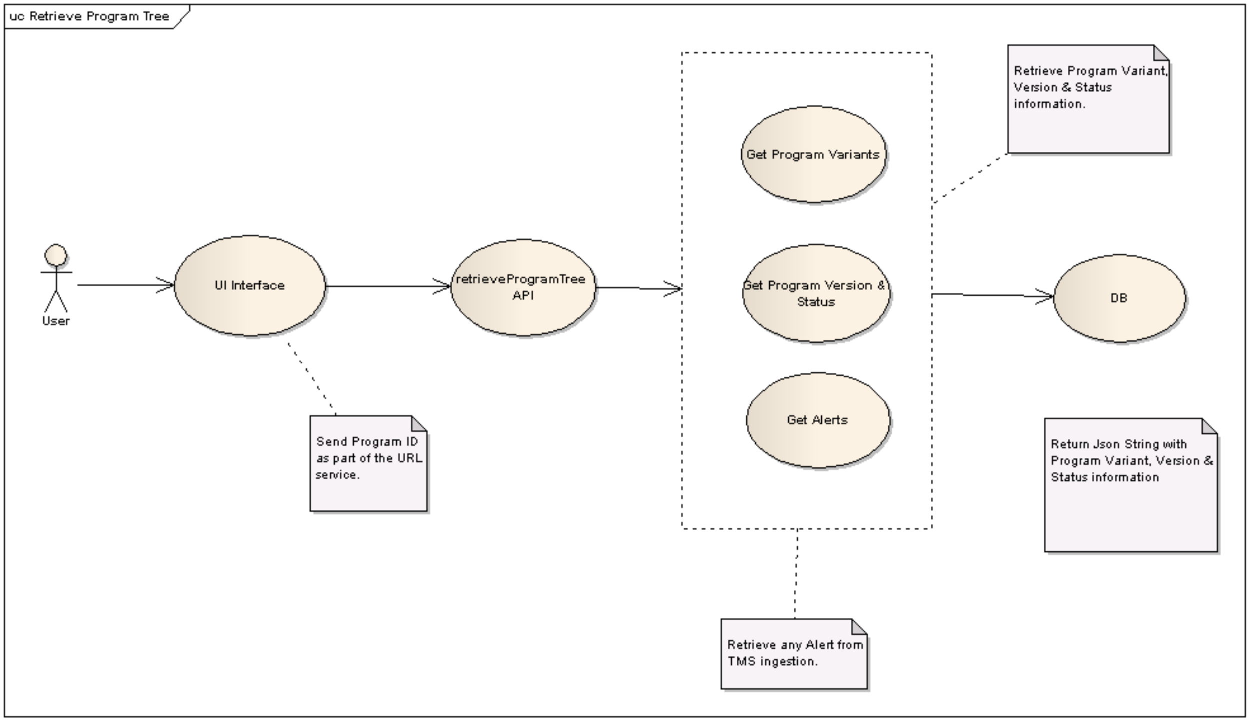
The following HTTP Headers are required to complete Request process.

DB FUNCTION RESPONSE CODE	RESPONSE CODE TO UI	DESCRIPTION
1	2103	Cannot pass DELETE status as Parameter, use operationType Parameter instead.
2	2103	Already set to ACTIVE status.
3	2103	Already set to APPROVED status.
4	2103	Cannot change status, current DELETE status.
5	2103	Cannot change status, there is no Program with APPROVED status.
6	2103	Status type should be change between APPROVED and ACTIVE status.
7	2103	Program does not exist.
8	2103	There is a version assigned with ACTIVE status, it cannot be deleted the Program at all.
9	2103	Operation does not allowed.
10	**0**	**OK completed**
11	2103	Oracle PL/SQL Error
12	2103	Current Status NEW -> ACTIVE is not valid, it must be APPROVED -> ACTIVE.
13	2103	Already set to REVIEWED status.
14	2103	Cannot change to REVIEWED status. Current status is either APPROVED or ACTIVED.
15	2103	TMS Program status should be either APPROVED or REVIEWED only.
N/A	2103	General Error

2.6 *Retrieve Program Status Tree*

2.6.1 Use Case Diagrams

Retrieve Program Status Tree

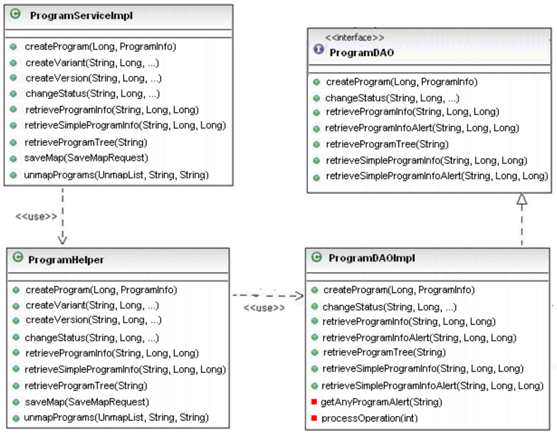

uc Retrieve Program Tree
User
UI Interface
retrieveProgramTree API
Send Program ID as part of the URL service.
Get Program Variants
Get Program Version & Status
Get Alerts
Retrieve Program Variant, Version & Status information.
DB
Return Json String with Program Variant, Version & Status information
Retrieve any Alert from TMS ingestion.

2.6.2 Tables

Below is a snapshot of how the table will look like in order to retrieve program status tree info.

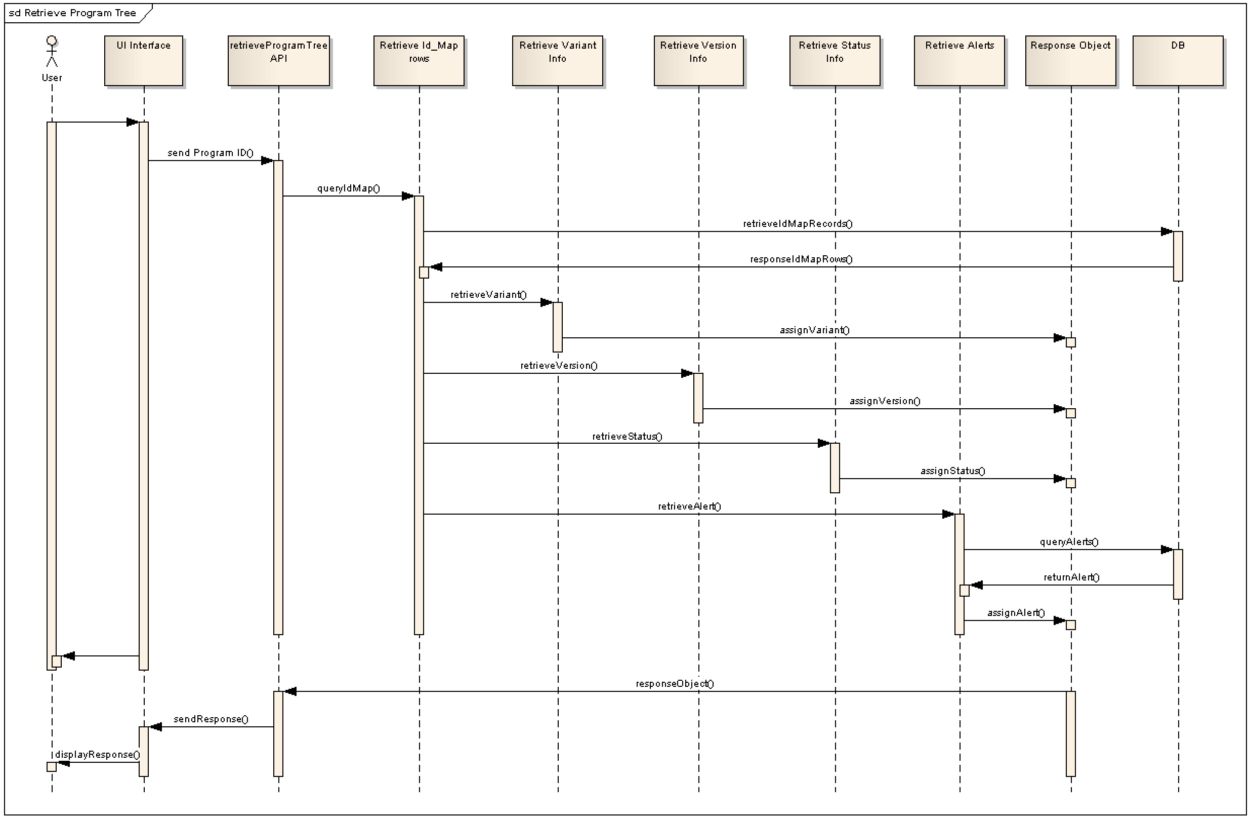
ID_MAP_REC	ROOT_ID	PROGRAM_ID	VARIANT_ID	STATUS	PROGRAM_VERSION	PGM_META_ID	OWNER	PPRFID
699380	NULL	SH005000210000	0	2	0	3382201	1	1281
699381	NULL	SH005000210000	0	2	0	3382202	1	1281
699382	NULL	SH005000210000	0	11	1	3382203	1	1244
699383	NULL	SH005000210000	0	11	1	3382204	1	1244
699384	NULL	SH005000210000	1	3	0	3382205	1	1275
699385	NULL	SH005000210000	1	3	0	3382206	1	1275

2.6.3 Class Diagrams


ProgramServiceImpl
createProgram(Long, ProgramInfo)
createVariant(String, Long, ...)
createVersion(String, Long, ...)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveProgramTree(String)
saveMap(SaveMapRequest)
unmapPrograms(UnmapList, String, String)
<<interface>>
ProgramDAO
createProgram(Long, ProgramInfo)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveProgramInfoAlert(String, Long, Long)
retrieveProgramTree(String)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveSimpleProgramInfoAlert(String, Long, Long)
<<use>>
ProgramHelper
createProgram(Long, ProgramInfo)
createVariant(String, Long, ...)
createVersion(String, Long, ...)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveProgramTree(String)
saveMap(SaveMapRequest)
unmapPrograms(UnmapList, String, String)
<<use>>
ProgramDAOImpl
createProgram(Long, ProgramInfo)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveProgramInfoAlert(String, Long, Long)
retrieveProgramTree(String)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveSimpleProgramInfoAlert(String, Long, Long)
getAnyProgramAlert(String)
processOperation(int)

2.6.4 Sequence Diagrams

Retrieve Program Status Tree



2.6.5 Core Components

The following core components are created:

- **ProgramServiceImpl.java** – Contains the Restful service available to use.
- **ProgramHelper.java** – Helper class.
- **ProgramDAO.java** – Interface.
- **ProgramDAOImpl.java** – Compose Program object.
- **ProgramUtils.java** – Class that contains useful methods.

2.6.6 Test Cases

- Retrieve Program Status Tree.
  - Set Program ID as URL parameter
  - Invoke retrieveProgramTree Api.
  - Retrieve Program Variant, Version & Status info.
  - Build Response Object.
  - Validate Response Object.

2.6.7 Request Action

The following URL may be used to retrieve program status tree:
http://*host:port*/ProgramTMS/resources/program/retrieveProgramTree/programId**={programId}**

PARAMETER	DESCRIPTION
{programId}	Valid Program Id – Alphanumeric value.

Example:
http://*host:port*/ProgramTMS/resources/program/retreiveProgramTree/programId=**SH005000440000**

HTTP Headers

The following HTTP Headers are required to complete Request process.

HTTP HEADER	VALUE
Content-Type	application/json
Accept	application/json

HTTP Method

The Request Action needs to define a HTTP **Get** method.

2.6.8 Response Action

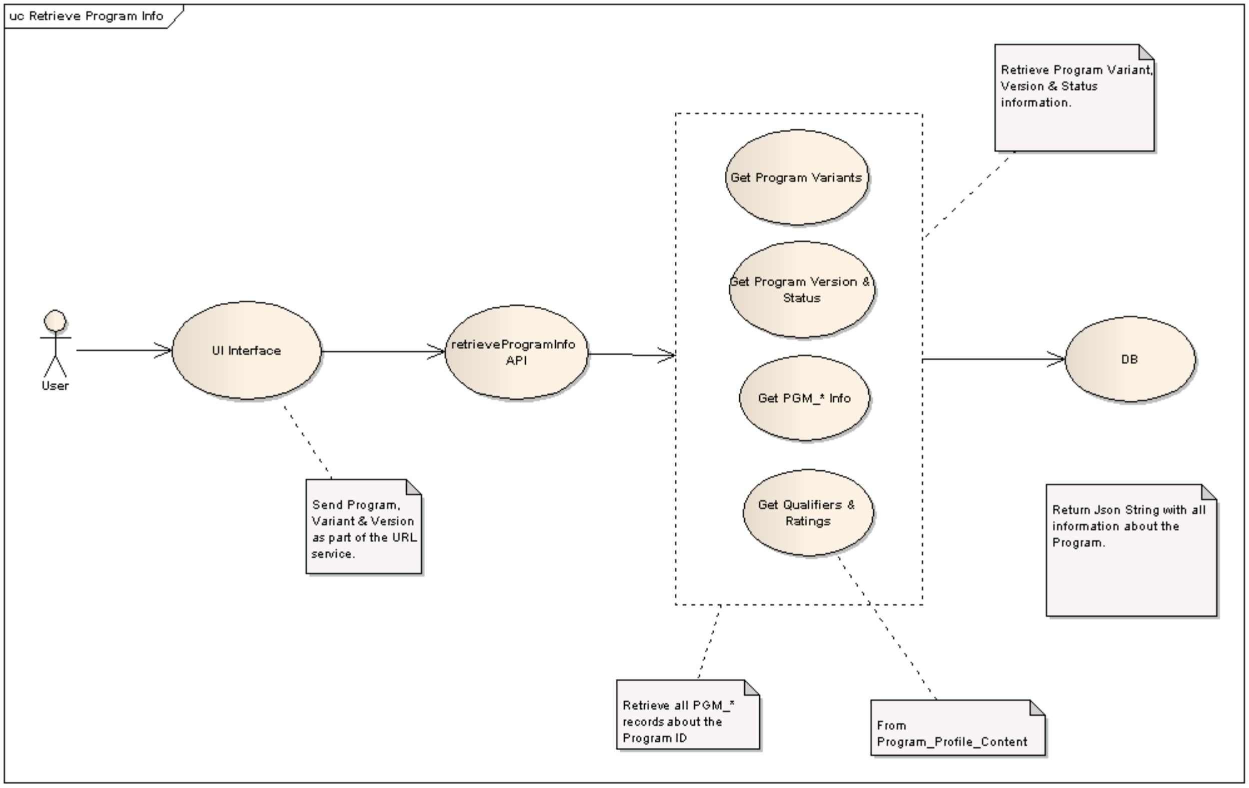
The following file describes a Response Object as example:

```
[{
   "programId": "SH005000010000",
   "tmsId": "SH005000010000",
   "programTypeId": 5,
   "programTypeDesc": "Series",
   "programPrefixTypeId": 5,
   "programPrefixTypeDesc": "SH",
   "variants":    [
            {
         "variantId": 0,
         "variantDesc": "Variant : 0",
         "versions": [         {
            "versionId": 1,
            "description": "Version : 1",
            "statusId": 11,
            "statusName": "APPROVED",
            "version": []
         }]
      },
            {
         "variantId": 2,
         "variantDesc": "This variant for Super Mario Bros",
         "versions": [         {
            "versionId": 0,
            "description": "Version : 0",
            "statusId": 11,
            "statusName": "APPROVED",
            "version": []
         }]
      }
   ],
   "alert": null
}]
```

2.7 *Retrieve Program Information (Complete model)*

2.7.1 Use Case Diagrams

Retrieve Program Info

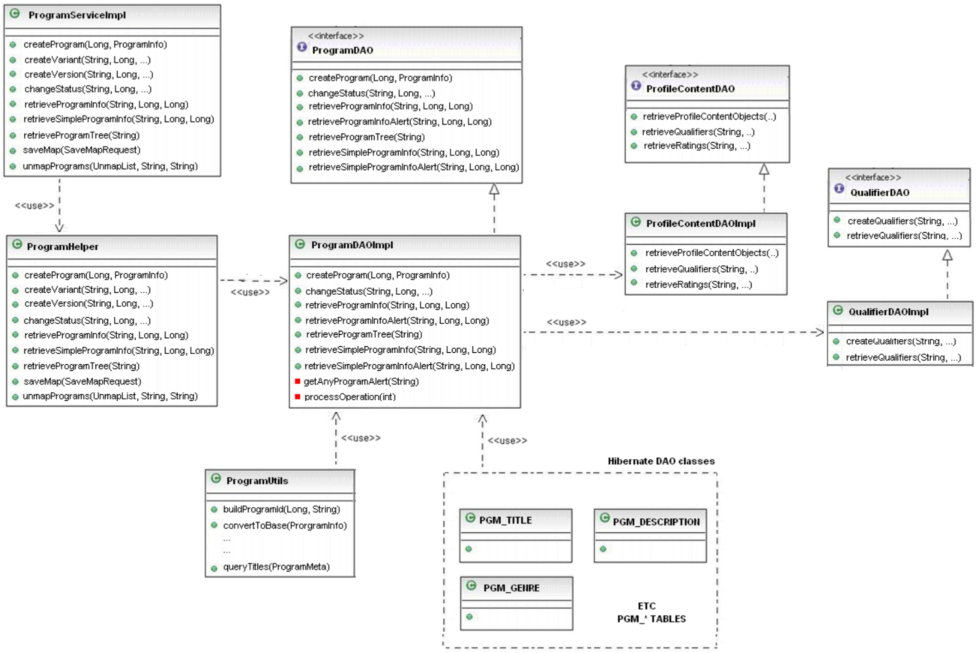

uc Retrieve Program Info
User
UI Interface
retrieveProgramInfo API
Get Program Variants
Get Program Version & Status
Get PGM_* Info
Get Qualifiers & Ratings
DB
Retrieve Program Variant, Version & Status information.
Send Program, Variant & Version as part of the URL service.
Return Json String with all information about the Program.
Retrieve all PGM_* records about the Program ID
From Program_Profile_Content

2.7.2 Tables

ID_MAP & PGM_* tables

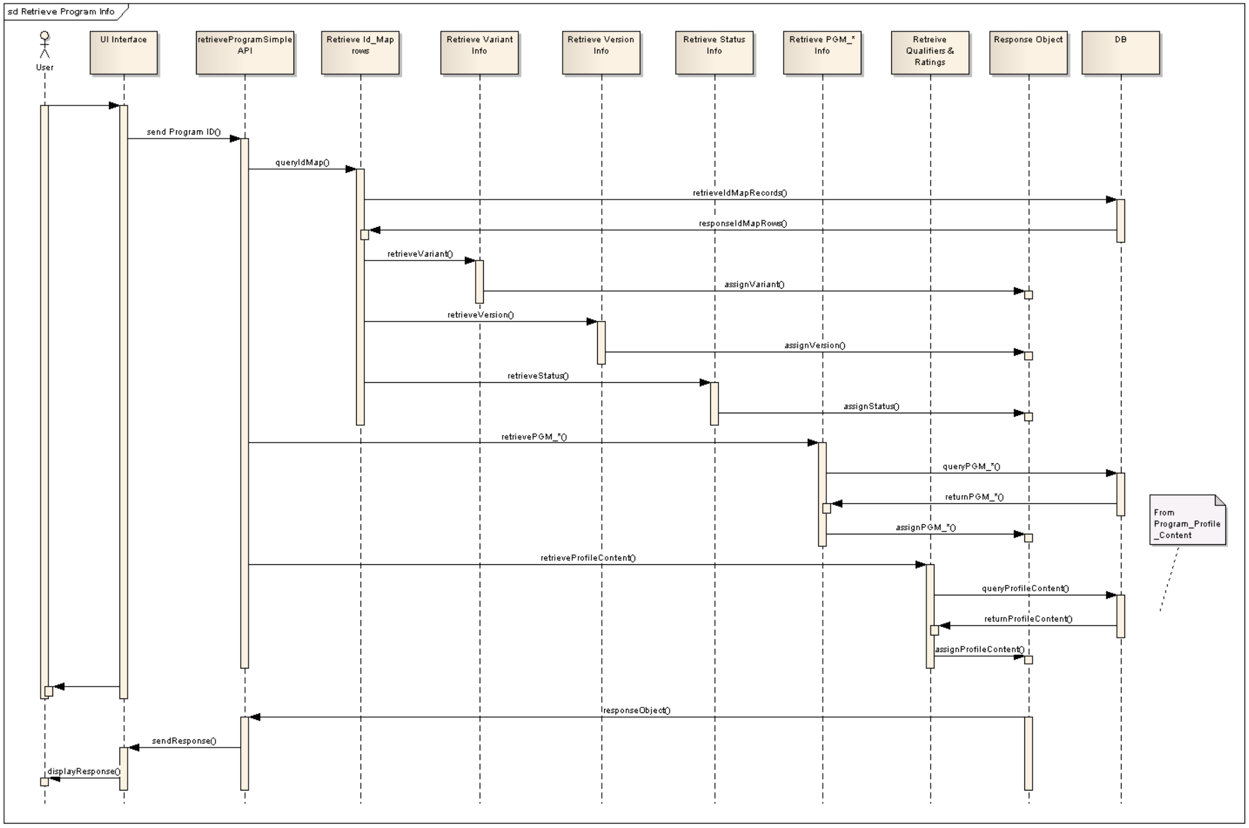
It will query ID_MAP table, plus PGM_TITLE, PGM_DESCRIPTION, PGM_GENRE, PGM_CREDIT, etc. tables.

2.7.3 Class Diagrams


ProgramServiceImpl
createProgram(Long, ProgramInfo)
createVariant(String, Long, ...)
createVersion(String, Long, ...)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveProgramTree(String)
saveMap(SaveMapRequest)
unmapPrograms(UnmapList, String, String)
<<interface>>
ProgramDAO
createProgram(Long, ProgramInfo)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveProgramInfoAlert(String, Long, Long)
retrieveProgramTree(String)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveSimpleProgramInfoAlert(String, Long, Long)
<<interface>>
ProfileContentDAO
retrieveProfileContentObjects(..)
retrieveQualifiers(String, ..)
retrieveRatings(String, ...)
<<interface>>
QualifierDAO
createQualifiers(String, ...)
retrieveQualifiers(String, ...)
<<use>>
ProgramHelper
createProgram(Long, ProgramInfo)
createVariant(String, Long, ...)
createVersion(String, Long, ...)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveProgramTree(String)
saveMap(SaveMapRequest)
unmapPrograms(UnmapList, String, String)
<<use>>
ProgramDAOImpl
createProgram(Long, ProgramInfo)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveProgramInfoAlert(String, Long, Long)
retrieveProgramTree(String)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveSimpleProgramInfoAlert(String, Long, Long)
getAnyProgramAlert(String)
processOperation(int)
<<use>>
ProfileContentDAOImpl
retrieveProfileContentObjects(..)
retrieveQualifiers(String, ..)
retrieveRatings(String, ...)
<<use>>
QualifierDAOImpl
createQualifiers(String, ...)
retrieveQualifiers(String, ...)
<<use>>
<<use>>
Hibernate DAO classes
ProgramUtils
buildProgramId(Long, String)
convertToBase(ProrgramInfo)
...
...
queryTitles(ProgramMeta)
PGM_TITLE
PGM_DESCRIPTION
PGM_GENRE
ETC
PGM_* TABLES

2.7.4 Sequence Diagrams

Retrieve Program Info



2.7.5 Core Components

The following core components are created:

- **ProgramServiceImpl.java** – Contains the Restful service available to use.
- **ProgramHelper.java** – Helper class.
- **ProgramDAO.java** – Interface.
- **ProgramDAOImpl.java** – Compose Program object.
- **CreateObjectDAO.java** – Interface.
- **CreateObjectDAOImpl.java** – Generic class to create the Program object.
- **ProgramUtils.java** – Class that contains useful methods.
- **ProfileContentDAO.java** – Interface.
- **ProfileContentDAOImpl.java** – Class to manage Qualifiers & Ratings for Profile Content Table.
- **QualifierDAO.java** – Interface.
- **QualifierDAOImpl.java** – Class to handle Qualifiers.
- **PgmTitle.java** – Class for PGM Title table.
- **PgmDescription.java** – Class for PGM Description table.
- **PgmGenre.java** – Class for PGM Genre table.
- **PgmCredits.java** – Class for PGM Credits table.
- **PgmSportsInfo.java** – Class for PGM Sports Info table.
- **PgmTrailer.java** – Class for PGM Trailer table.
- **Pgm*.java** – Class for rest of PGM tables.

2.7.6 Test Cases

- Retrieve Program Information
  - Provide Program, Variant & Version IDs as part of the URL service.
  - Invoke retrieveProgramInfo API.
  - Retrieve Program Variant, Version & Status info.
  - Retrieve all PGM_* info about the Program such as Title, Description, Genre, etc.
  - Retrieve Qualifiers and Ratings [Schedule Ratings].
  - Build Response Object.
  - Validate Response Object.

2.7.7 Request Action

The following URL may be used to retrieve Program Information:
http://*host:port*/ProgramTMS/resources/program/programInfo/**{programId}**&**{variantId}**&**{versionId}**

PARAMETER	DESCRIPTION
{programId}	Valid Program Id – Alphanumeric value.
{variantId}	Valid Variant Id – Numeric value.
{versionId}	Valid Version Id – Numeric value.

Example:
http://*host:port*/ProgramTMS/resources/program/programInfo/ **SH005000010000**&**0**&**0**

Retrieve Program Info with ACTIVE status or latest NEW status.

The following URL will retrieve ONE and ONLY ONE Program/Variant object, in case there are several versions available. The purpose of this, it's to retrieve ONLY the object with ACTIVE status [High Priority over other objects with NEW status] or the latest object NEW status in case there is NO object with ACTIVE status present.

Example:
http://*host:port*/ProgramTMS/resources/program/programInfo/ **SH005000010000**&**0**&**-3**

HTTP Headers

The following HTTP Headers are required to complete Request process.

HTTP HEADER	VALUE
Content-Type	application/json
Accept	application/json

HTTP Method

The Request Action needs to define a HTTP **Get** method.

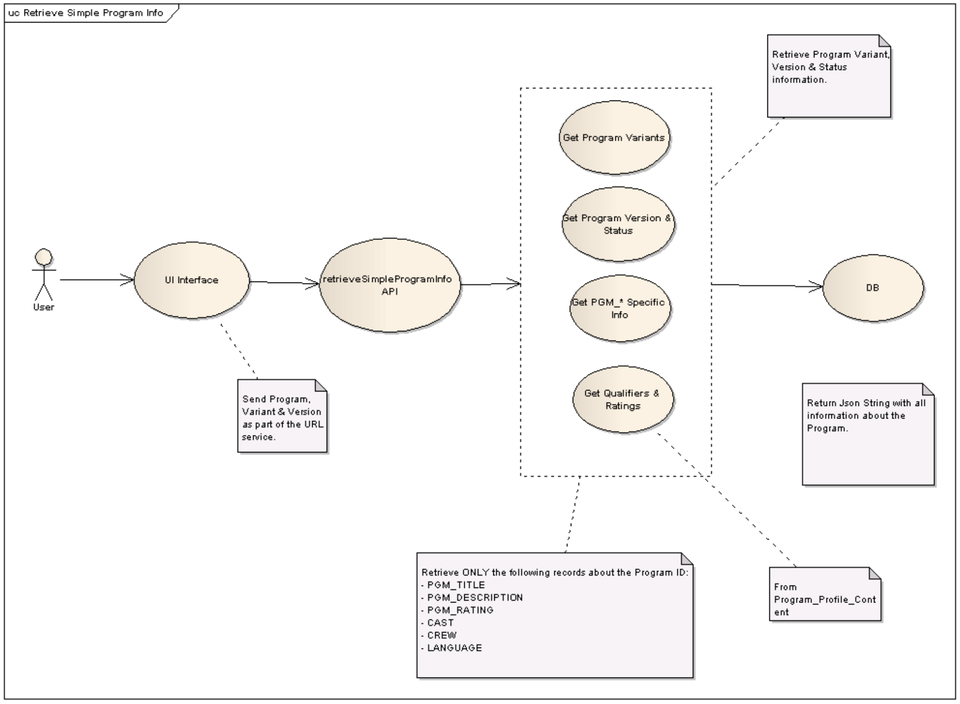
2.7.8 Response Action

The following file describes a Response Object as example:
http://*host:port*/ProgramTMS/

2.8 *Retrieve Program Information (Simple model)*

2.8.1 Use Case Diagrams

Retrieve Simple Program Info

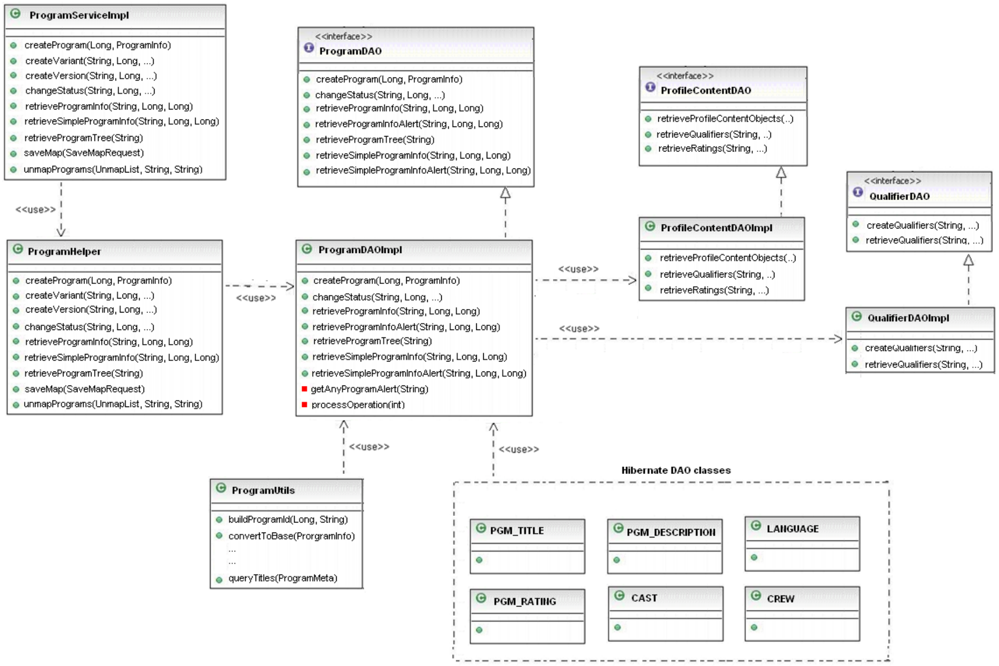

uc Retrieve Simple Program Info
Retrieve Program Variant, Version & Status information.
Get Program Variants
Get Program Version & Status
User
UI Interface
retrieveSimpleProgramInfo API
Get PGM_* Specific Info
DB
Get Qualifiers & Ratings
Send Program, Variant & Version as part of the URL service.
Return Json String with all information about the Program.
Retrieve ONLY the following records about the Program ID:
- PGM_TITLE
- PGM_DESCRIPTION
- PGM_RATING
- CAST
- CREW
- LANGUAGE
From Program_Profile_Content

2.8.2 Tables

ID_MAP & PGM_* tables

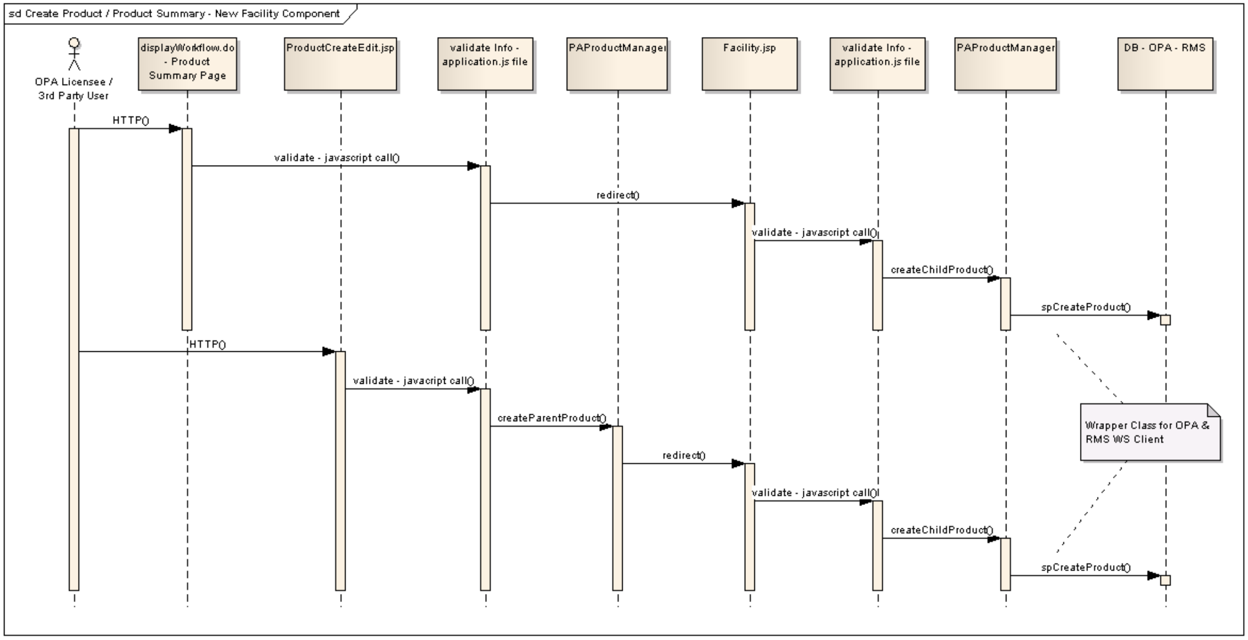
It will query ID_MAP table and ONLY the following information PGM_TITLE, PGM_DESCRIPTION, PGM_RATING, CAST, CREW, LANGUAGE & PROFILE_CONTENT.

2.8.3 Class Diagrams


ProgramServiceImpl
createProgram(Long, ProgramInfo)
createVariant(String, Long, ...)
createVersion(String, Long, ...)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveProgramTree(String)
saveMap(SaveMapRequest)
unmapPrograms(UnmapList, String, String)
<<interface>>
ProgramDAO
createProgram(Long, ProgramInfo)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveProgramInfoAlert(String, Long, Long)
retrieveProgramTree(String)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveSimpleProgramInfoAlert(String, Long, Long)
<<interface>>
ProfileContentDAO
retrieveProfileContentObjects(..)
retrieveQualifiers(String, ..)
retrieveRatings(String, ...)
<<interface>>
QualifierDAO
createQualifiers(String, ...)
retrieveQualifiers(String, ...)
<<use>>
ProgramHelper
createProgram(Long, ProgramInfo)
createVariant(String, Long, ...)
createVersion(String, Long, ...)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveProgramTree(String)
saveMap(SaveMapRequest)
unmapPrograms(UnmapList, String, String)
<<use>>
ProgramDAOImpl
createProgram(Long, ProgramInfo)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveProgramInfoAlert(String, Long, Long)
retrieveProgramTree(String)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveSimpleProgramInfoAlert(String, Long, Long)
getAnyProgramAlert(String)
processOperation(int)
<<use>>
ProfileContentDAOImpl
retrieveProfileContentObjects(..)
retrieveQualifiers(String, ..)
retrieveRatings(String, ...)
<<use>>
QualifierDAOImpl
createQualifiers(String, ...)
retrieveQualifiers(String, ...)
<<use>>
<<use>>
ProgramUtils
buildProgramId(Long, String)
convertToBase(ProrgramInfo)
...
...
queryTitles(ProgramMeta)
Hibernate DAO classes
PGM_TITLE
PGM_DESCRIPTION
LANGUAGE
PGM_RATING
CAST
CREW

2.8.4 Sequence Diagrams

[Create Product - Product Summary] -- Create Facility Component


sd Create Product / Product Summary - New Facility Component
OPA Licensee / 3rd Party User
displayWorkflow.do - Product Summary Page
ProductCreateEdit.jsp
validate Info - application.js file
PAProductManager
Facility.jsp
validate Info - application.js file
PAProductManager
DB - OPA - RMS
HTTP()
validate - javascript call()
redirect()
validate - javascript call()
createChildProduct()
spCreateProduct()
HTTP()
validate - javacript call()
createParentProduct()
redirect()
validate - javascript call()
createChildProduct()
spCreateProduct()
Wrapper Class for OPA & RMS WS Client

2.8.5 Core Components

The following core components are created:

- **ProgramServiceImpl.java** – Contains the Restful service available to use.
- **ProgramHelper.java** – Helper class.
- **ProgramDAO.java** – Interface.
- **ProgramDAOImpl.java** – Compose Program object.
- **CreateObjectDAO.java** – Interface.
- **CreateObjectDAOImpl.java** – Generic class to create the Program object.
- **ProgramUtils.java** – Class that contains useful methods.
- **ProfileContentDAO.java** – Interface.
- **ProfileContentDAOImpl.java** – Class to manage Qualifiers & Ratings for Profile Content Table.
- **QualifierDAO.java** – Interface.
- **QualifierDAOImpl.java** – Class to handle Qualifiers.
- **PgmTitle.java** – Class for PGM Title table.
- **PgmDescription.java** – Class for PGM Description table.
- **PgmRating.java** – Class for PGM Genre table.
- **PgmCredits.java** – Class for PGM Credits table for [CAST & CREW].

2.8.6 Test Cases

- Retrieve Program Information
  - Provide Program, Variant & Version IDs as part of the URL service.
  - Invoke retrieveSimpleProgramInfo API.
  - Retrieve Program Variant, Version & Status info.
  - Retrieve ONLY Title, Description, Cast, Crew & Language info.
  - Retrieve Qualifiers and Ratings [Schedule Ratings].
  - Build Response Object.
  - Validate Response Object.

2.8.7 Request Action

The following URL may be used to retrieve Simple Program Information:
http://*host:port*/ProgramTMS/resources/program/programSimpleInfo/**{programId}**&**{variantId}**&**{versionId}**

PARAMETER	DESCRIPTION
{programId}	Valid Program Id – Alphanumeric value.
{variantId}	Valid Variant Id – Numeric value.
{versionId}	Valid Version Id – Numeric value.

Example:
http://*host:port*/ProgramTMS/resources/program/programSimpleInfo/ **SH005000010000**&**0**&**0**

Retrieve Simple Program Info with ACTIVE status or latest NEW status.

The following URL will retrieve ONE and ONLY ONE Program/Variant object, in case there are several versions available. The purpose of this, it's to retrieve ONLY the object with ACTIVE status [High Priority over other objects with NEW status] or the latest object NEW status in case there is NO object with ACTIVE status present.

Example:
http://*host:port*/ProgramTMS/resources/program/programSimpleInfo/ **SH005000010000**&**0**&**-3**

HTTP Headers

The following HTTP Headers are required to complete Request process.

HTTP HEADER	VALUE
Content-Type	application/json
Accept	application/json

HTTP Method

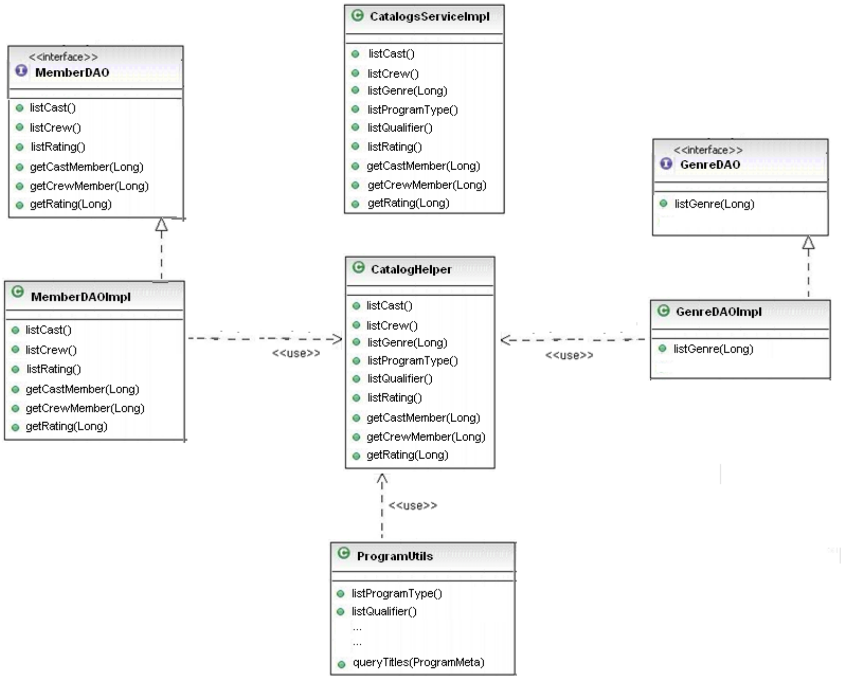
The Request Action needs to define a HTTP **Get** method.

2.8.8 Response Action

The following file describes a Response Object as example:
http://*host:port*/ProgramTMS/

2.9 ***Listing Catalogs***

2.9.1 Class Diagrams



2.9.2 Core Components

The following core components are created:

- **CatalogServiceImpl.java** – Contains the Restful service available to use.
- **CatalogHelper.java** – Helper class.
- **MemberDAO.java** – Interface.
- **MemberDAOImpl.java** – Generic class to retrieve Cast, Crew & Rating objects.
- **ProgramUtils.java** – Class that contains useful methods like to retrieve Program Type & Qualifier records.
- **GenreDAO.java** – Interface.
- **GenreDAOImpl.java** – Generic class to retrieve Genre object.

2.9.3 Response Action

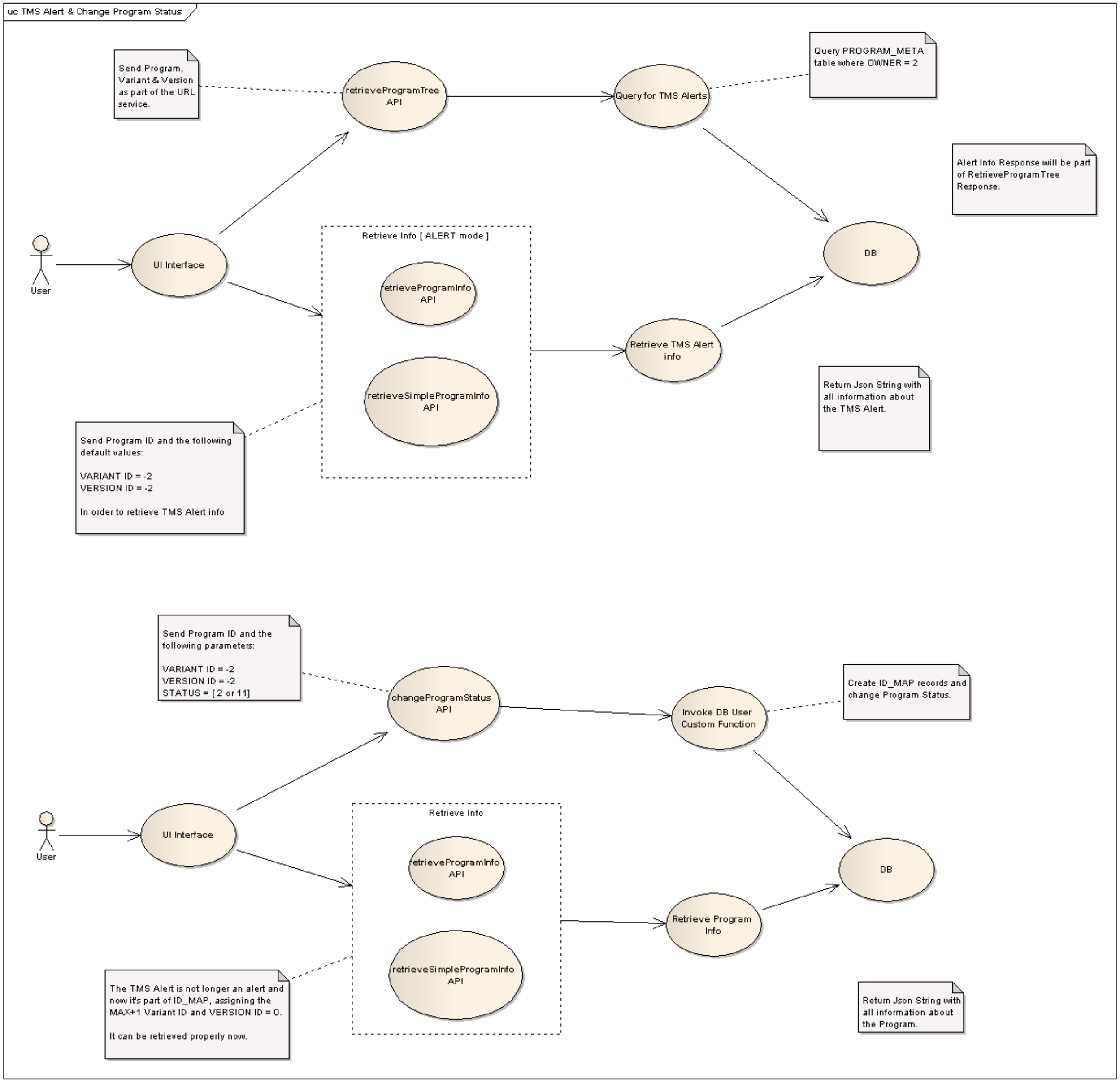
The following file describes a Response Object as example:

List Cast
http://*host:port*/ProgramTMS/
List Crew
http://*host:port*/ProgramTMS/
List Genre
http://*host:port*/ProgramTMS/
List Program Type
http://*host:port*/ProgramTMS/
List Rating
http://*host:port*/ProgramTMS/
List Qualifier
http://*host:port*/ProgramTMS/

2.10 *TMS Alert & Change Program Status*

2.10.1 Use Case Diagrams

TMS Alert & Change Program Status



Send Program, Variant & Version as part of the URL service.

Query PROGRAM_META table where OWNER = 2

Alert Info Response will be part of RetrieveProgramTree Response.

Return Json String with all information about the TMS Alert.

Send Program ID and the following default values:

VARIANT ID = -2
VERSION ID = -2

In order to retrieve TMS Alert info

Send Program ID and the following parameters:

VARIANT ID = -2
VERSION ID = -2
STATUS = [2 or 11]

Create ID_MAP records and change Program Status.

The TMS Alert is not longer an alert and now it's part of ID_MAP, assigning the MAX+1 Variant ID and VERSION ID = 0.

It can be retrieved properly now.

Return Json String with all information about the Program.

2.10.2 Tables

TMS ALERT FOUND.

Id_Map

Below is a snapshot of how the table will look like when TMS Alert is not part of it.

ID_MAP_REC	ROOT_ID	PROGRAM_ID	VARIANT_ID	STATUS	PROGRAM_VERSION	PGM_META_ID	OWNER	PPRFID

Id_Map_Log

Below is a snapshot of how the table will look like when TMS Alert is not part of it.

ID	ID_MAP_REC	USER_ID	MODIFY_DATE

Program_Meta

Below is a snapshot of how the table will look like after a TMS ALERT is present.

PGM_META_ID	VERSION	PROGRAM_ID	LANG_ID	ROOT_ID	OWNER	TYPE	…	ADULT
3382228	0	SH005081010000	1	NULL	**2**	5	…	NULL
3382229	0	SH005081010000	2	NULL	**2**	5	…	NULL

NOW TMS ALERT IS EITHER REVIEWED OR APPROVED

Id_Map

Below is a snapshot of how the table will look like after a new version is created.

ID_MAP_REC	ROOT_ID	PROGRAM_ID	VARIANT_ID	STATUS	PROGRAM_VERSION	PGM_META_ID	OWNER	PPRFID
699380	NULL	SH005081010000	0	3	0	3382201	1	1281
699381	NULL	SH005081010000	0	3	0	3382202	1	1281
699415	**NULL**	**SH005081010000**	**1**	**11**	**0**	**3382228**	**1**	**Null**
699416	**NULL**	**SH005081010000**	**1**	**11**	**0**	**3382229**	**1**	**Null**

Id_Map_Log

Below is a snapshot of how the table will look like after a new version is created.

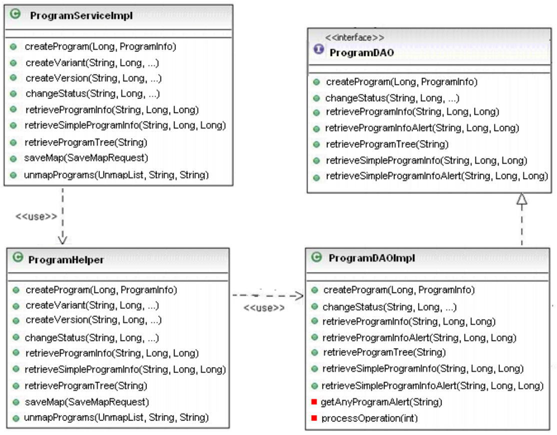
ID	ID_MAP_REC	USER_ID	MODIFY_DATE
23915	699415	50	21-FEB-12 17.15.24
23916	699416	50	21-FEB-12 17.15.25

Program_Meta

Below is a snapshot of how the table will look like after a new version is created.

PGM_META_ID	VERSION	PROGRAM_ID	LANG_ID	ROOT_ID	OWNER	TYPE	…	ADULT
3382228	0	SH005000210000	1	NULL	**2**	5	…	NULL
3382229	0	SH005000210000	2	NULL	**2**	5	…	NULL

2.10.3 Class Diagrams


ProgramServiceImpl
createProgram(Long, ProgramInfo)
createVariant(String, Long, ...)
createVersion(String, Long, ...)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveProgramTree(String)
saveMap(SaveMapRequest)
unmapPrograms(UnmapList, String, String)
<<interface>>
ProgramDAO
createProgram(Long, ProgramInfo)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveProgramInfoAlert(String, Long, Long)
retrieveProgramTree(String)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveSimpleProgramInfoAlert(String, Long, Long)
<<use>>
ProgramHelper
createProgram(Long, ProgramInfo)
createVariant(String, Long, ...)
createVersion(String, Long, ...)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveProgramTree(String)
saveMap(SaveMapRequest)
unmapPrograms(UnmapList, String, String)
<<use>>
ProgramDAOImpl
createProgram(Long, ProgramInfo)
changeStatus(String, Long, ...)
retrieveProgramInfo(String, Long, Long)
retrieveProgramInfoAlert(String, Long, Long)
retrieveProgramTree(String)
retrieveSimpleProgramInfo(String, Long, Long)
retrieveSimpleProgramInfoAlert(String, Long, Long)
getAnyProgramAlert(String)
processOperation(int)

2.10.4 Core Components

The following core components are created:

- **ProgramServiceImpl.java** – Contains the Restful service available to use.
- **ProgramHelper.java** – Helper class.
- **ProgramDAO.java** – Interface.
- **ProgramDAOImpl.java** – Compose Program object.
- **ProgramUtils.java** – Class that contains useful methods.

2.10.5 Test Cases

- TMS Alert & Change Program Status
  - Provide Program to invoke retriveProgramTree API.
  - Display Variant(s) & Version(s) related to the Program ID.
  - Display basic ALERT info as part of the Response object.
  - Retrieve ALERT info [Complete] by invoking retrieveProgramInfo API passing ProgramID, VariantID = -2 & VersionID = -2 parameters.
  - In order to get ALERT as part of ID_MAP, invoke changeProgramStatus API by passing ProgramID, Status = [2 or 11], VariantID = -2 & VersionID = -2 parameters.
  - ALERT is turn off now and it's part of ID_MAP.
  - Invoke retrieveProgramTree API to confirm that the MAX+1 VARIANT ID value is the ALERT.

2.10.6 Request & Response Action

Retrieve Program Tree in order to find any TMS Alert.

The following URL may be used to retrieve Program Tree Information:

Example Request:
http://*host:port*/ProgramTMS/resources/program/retreiveProgramTree/programId=**SH005081010000**

Example Response [Alert found]:

```
[{
   "programId": "SH005081010000",
   "tmsId": "SH005081010000",
   "programTypeId": 5,
   "programTypeDesc": "Series",
   "programPrefixTypeId": 5,
   "programPrefixTypeDesc": "SH",
   "variants":    [
            {
         "variantId": 0,
         "variantDesc": "This variant for Super Mario Bros",
         "versions": [         {
            "versionId": 0,
            "description": "Version : 0",
            "statusId": 1,
            "statusName": "NEW",
            "version": []
         }]
      },
            {
         "variantId": 1,
         "variantDesc": "This variant 1 for The Legend of Zelda",
         "versions": [         {
            "versionId": 0,
            "description": "Version : 0",
            "statusId": 3,
            "statusName": "ACTIVED",
            "version": []
         }]
      }
   ],
   "alert":    {
      "variantId": 2,
      "versionId": 0,
      "statusId": 1,
      "statusName": "NEW",
      "pgmMetaId":       [
         3443961,
         3443962
      ]
   }
}]
```

Retrieve Program Information about the TMS Alert.

The following URL may be used to retrieve Program Information:

Example Request:
http://*host:port*/ProgramTMS/resources/program/programInfo/ **SH005000010000**&**-2**&**-2**

Important
In order to retrieve TMS Alert Info
Variant ID = -2
Version ID = -2

Change TMS Alert status to either REVIEWED or APPROVED.

The following URL may be used to change TMS Alert status to either REVIEWED or APPROVED:

Example Request:
http://*host:port*/ProgramTMS/resources/program/changeProgramStatus/programId= **SH005000010000**&variantId=**-2**&versionId=**-2**&statusId=11&userId=50

Example Response:

```
{
    "statusCode": 0,
    "statusDescription": "OK completed"
}
```

Retrieve Program Tree in order to identify TMS Alert as part of the Program Tree.

The following URL may be used to retrieve Program Tree Information and the TMS Alert is not longer available:

Example Request:
http://*host:port*/ProgramTMS/resources/program/retreiveProgramTree/programId=**SH005081010000**

Example Response [Alert is not longer available]:

```
[{
   "programId": "SH005081010000",
   "tmsId": "SH005081010000",
   "programTypeId": 5,
   "programTypeDesc": "Series",
   "programPrefixTypeId": 5,
   "programPrefixTypeDesc": "SH",
   "variants":    [
            {
         "variantId": 0,
         "variantDesc": "This variant for Super Mario Bros",
         "versions": [         {
            "versionId": 0,
            "description": "Version : 0",
            "statusId": 1,
            "statusName": "NEW",
            "version": []
         }]
      },
            {
         "variantId": 1,
         "variantDesc": "This variant 1 for The Legend of Zelda",
         "versions": [         {
            "versionId": 0,
            "description": "Version : 0",
            "statusId": 3,
            "statusName": "ACTIVED",
            "version": []
         }]
      },
            {
         "variantId": 2,
         "variantDesc": "Variant : 2",
         "versions": [         {
            "versionId": 0,
            "description": "Version : 0",
            "statusId": 11,
            "statusName": "APPROVED",
            "version": []
         }]
      }
   ],
   "alert": null
}]
```

2.11 *Update Variant Name*

2.11.1 Tables

Id_Map [Before Update]

Below is a snapshot of how the table will look like before updating variant name.

ID	PROGRAM_ID	VARIANT_ID	DESCRIPTION
699380	SH005000210000	1	This is my first Variant

Id_Map [After Update]

Below is a snapshot of how the table will look like after updating variant name.

ID	PROGRAM_ID	VARIANT_ID	DESCRIPTION
699380	SH005000210000	1	**This super variant is the best**

2.11.2 Core Components

The following core components are created:

- **ProgramServiceImpl.java** – Contains the Restful service available to use.
- **ProgramHelper.java** – Helper class.
- **VariantDAO.java** – Interface.
- **VariantDAOImpl.java** – Generic class to create the Program object.
- **ProgramUtils.java** – Class that contains useful methods.

2.11.3 Test Cases

- Update Variant Name.
  - Retrieve Variant Name Information
  - Build Json Object
  - Post Json Object into updateVariantName API.
  - Complete update process.
  - Log process.
  - Response process as completed.

2.11.4 Request Action

The following URL may be used to create a program:
http://*host:port*/ProgramTMS/resources/program/updateVariantName/programId=**{programId}**&variantId=**{variantId}**&userId=**{userId}**&variantName={variantName}

PARAMETER	DESCRIPTION
{programId}	Valid Program Id – Alphanumeric value.
{variantId}	Valid Variant Id – Numeric value.
{userId}	Valid User Id – Numeric value.
{variantName}	Variant Name – Alphanumeric value.

Example:
http://*host:port*/ProgramTMS/resources/program/updateVariantName/programId=**SH005000010000**&variantId=**0**&userId=**50**&variantName=This_super_variant

HTTP Headers

The following HTTP Headers are required to complete Request process.

HTTP HEADER	VALUE
Content-Type	application/json
Accept	application/json

HTTP Method

The Request Action needs to define a HTTP **Post** method.

2.11.5 Response Action

The following file describes a Response Object as example:

```
{
   "statusCode": 0,
   "statusDescription": "OK completed"
}
```

2.12 *Update IMDB*

2.12.1 Tables

IMDB [Before Update]

Below is a snapshot of how the table will look like before updating Imdb info.

ID	PROGRAM_ID	IMDB_ID	DESCRIPTION	…	IMDB_URL
21	SH005000210000	tt0111161	The Shawshank Redemption	…	3382201

IMDB [After Update]

Below is a snapshot of how the table will look like before updating Imdb info.

ID	PROGRAM_ID	IMDB_ID	DESCRIPTION	…	IMDB_URL
21	SH005000210000	**tt0111986**	**The Legend of Zelda**	**…**	**33825896**

2.12.2 Core Components

The following core components are created:

- **ProgramServiceImpl.java** – Contains the Restful service available to use.
- **ProgramHelper.java** – Helper class.
- **ProgramDAO.java** – Interface.
- **ProgramDAOImpl.java** – Compose Program object.

2.12.3 Test Cases

- Update IMDB.
  - Retrieve IMDB Information
  - Build Json Object
  - Post Json Object into updateImdb API.
  - Complete update process.
  - Log process.
  - Response process as completed.

2.12.4 Request Action

The following URL may be used to create a program:
http://*host:port*/ProgramTMS/resources/program/updateImdb/userId=**{userId}**

PARAMETER	DESCRIPTION
{userId}	Valid User Id – Numeric value.

Example:
http://*host:port*/ProgramTMS/resources/program/updateImdb/userId=50

HTTP Headers

The following HTTP Headers are required to complete Request process.

HTTP HEADER	VALUE
Content-Type	application/json
Accept	application/json

HTTP Method

The Request Action needs to define a HTTP **Post** method.

Request Object

The following file describes a Request Object as example:

```
{
   "programId" : "SH005000410000",
   "imdbId" : "tt0103210",
   "imdbUrl" : "http://www.imdb.com/title/tt0103210/",
   "description" : "The Legend of Zelda"
}
```

2.12.5 Response Action

The following file describes a Response Object as example:

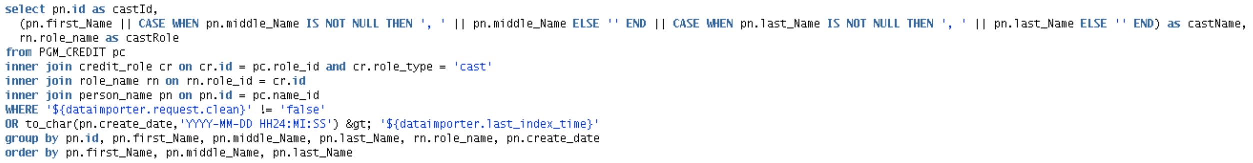
```
{
   "statusCode": 0,
   "statusDescription": "OK completed"
}
```

2.13 *Solr Searching*

2.13.1 SQL queries

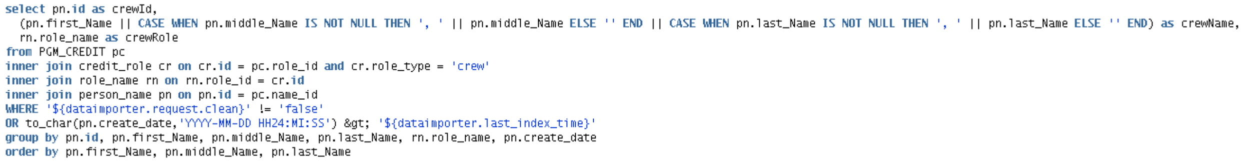
The following SQL queries will retrieve data to create SOLR indexes.

For Cast



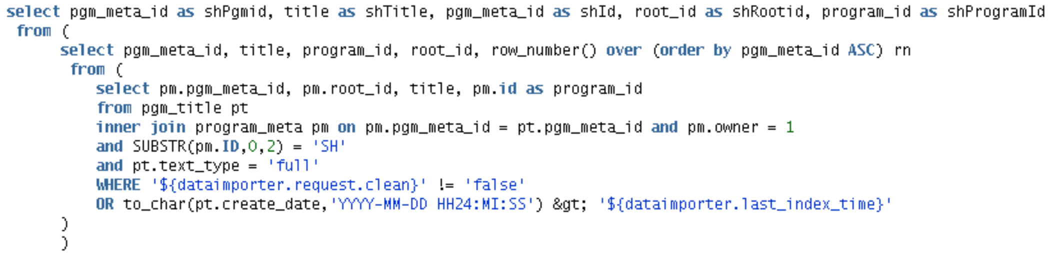
```
select pn.id as castId,
  (pn.first_Name || CASE WHEN pn.middle_Name IS NOT NULL THEN ', ' || pn.middle_Name ELSE '' END || CASE WHEN pn.last_Name IS NOT NULL THEN ', ' || pn.last_Name ELSE '' END) as castName,
  rn.role_name as castRole
from PGM_CREDIT pc
inner join credit_role cr on cr.id = pc.role_id and cr.role_type = 'cast'
inner join role_name rn on rn.role_id = cr.id
inner join person_name pn on pn.id = pc.name_id
WHERE '${dataimporter.request.clean}' != 'false'
OR to_char(pn.create_date,'YYYY-MM-DD HH24:MI:SS') &gt; '${dataimporter.last_index_time}'
group by pn.id, pn.first_Name, pn.middle_Name, pn.last_Name, rn.role_name, pn.create_date
order by pn.first_Name, pn.middle_Name, pn.last_Name
```

For Crew



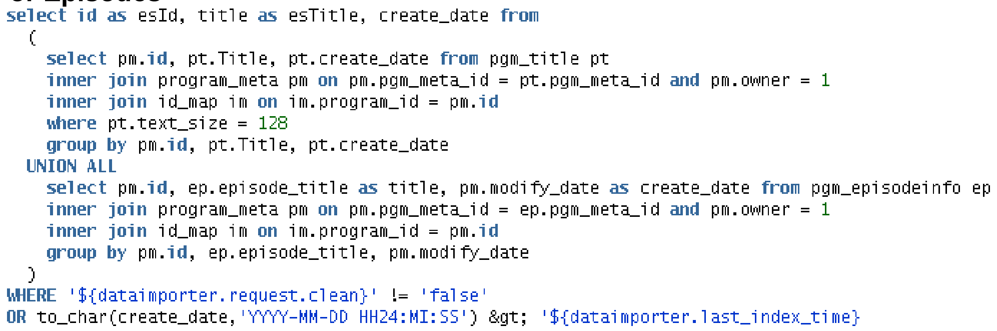
```
select pn.id as crewId,
  (pn.first_Name || CASE WHEN pn.middle_Name IS NOT NULL THEN ', ' || pn.middle_Name ELSE '' END || CASE WHEN pn.last_Name IS NOT NULL THEN ', ' || pn.last_Name ELSE '' END) as crewName,
  rn.role_name as crewRole
from PGM_CREDIT pc
inner join credit_role cr on cr.id = pc.role_id and cr.role_type = 'crew'
inner join role_name rn on rn.role_id = cr.id
inner join person_name pn on pn.id = pc.name_id
WHERE '${dataimporter.request.clean}' != 'false'
OR to_char(pn.create_date,'YYYY-MM-DD HH24:MI:SS') &gt; '${dataimporter.last_index_time}'
group by pn.id, pn.first_Name, pn.middle_Name, pn.last_Name, rn.role_name, pn.create_date
order by pn.first_Name, pn.middle_Name, pn.last_Name
```

For Series & Shows



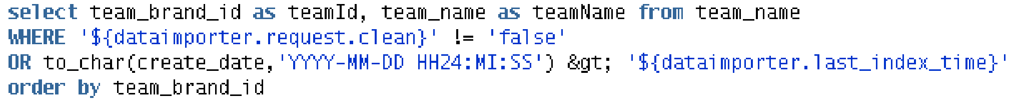
```
select pgm_meta_id as shPgmid, title as shTitle, pgm_meta_id as shId, root_id as shRootid, program_id as shProgramId
 from (
      select pgm_meta_id, title, program_id, root_id, row_number() over (order by pgm_meta_id ASC) rn
       from (
          select pm.pgm_meta_id, pm.root_id, title, pm.id as program_id
          from pgm_title pt
          inner join program_meta pm on pm.pgm_meta_id = pt.pgm_meta_id and pm.owner = 1
          and SUBSTR(pm.ID,0,2) = 'SH'
          and pt.text_type = 'full'
          WHERE '${dataimporter.request.clean}' != 'false'
          OR to_char(pt.create_date,'YYYY-MM-DD HH24:MI:SS') &gt; '${dataimporter.last_index_time}'
      )
      )
```

For Episodes



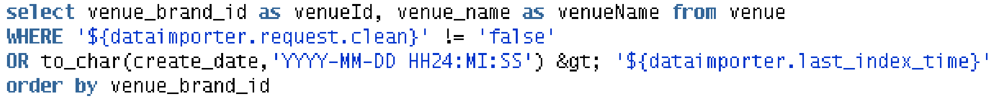
```
select id as esId, title as esTitle, create_date from
  (
    select pm.id, pt.Title, pt.create_date from pgm_title pt
    inner join program_meta pm on pm.pgm_meta_id = pt.pgm_meta_id and pm.owner = 1
    inner join id_map im on im.program_id = pm.id
    where pt.text_size = 128
    group by pm.id, pt.Title, pt.create_date
  UNION ALL
    select pm.id, ep.episode_title as title, pm.modify_date as create_date from pgm_episodeinfo ep
    inner join program_meta pm on pm.pgm_meta_id = ep.pgm_meta_id and pm.owner = 1
    inner join id_map im on im.program_id = pm.id
    group by pm.id, ep.episode_title, pm.modify_date
  )
WHERE '${dataimporter.request.clean}' != 'false'
OR to_char(create_date,'YYYY-MM-DD HH24:MI:SS') &gt; '${dataimporter.last_index_time}
```

For Teams



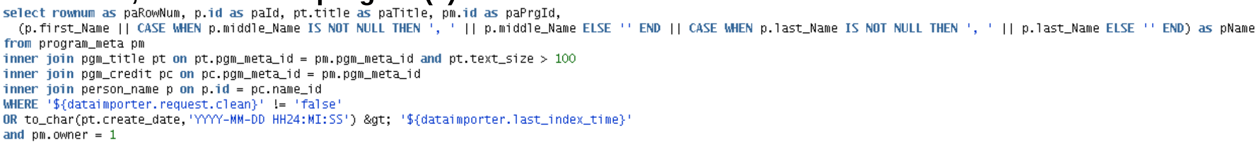
```
select team_brand_id as teamId, team_name as teamName from team_name
WHERE '${dataimporter.request.clean}' != 'false'
OR to_char(create_date,'YYYY-MM-DD HH24:MI:SS') &gt; '${dataimporter.last_index_time}'
order by team_brand_id
```

For Venues



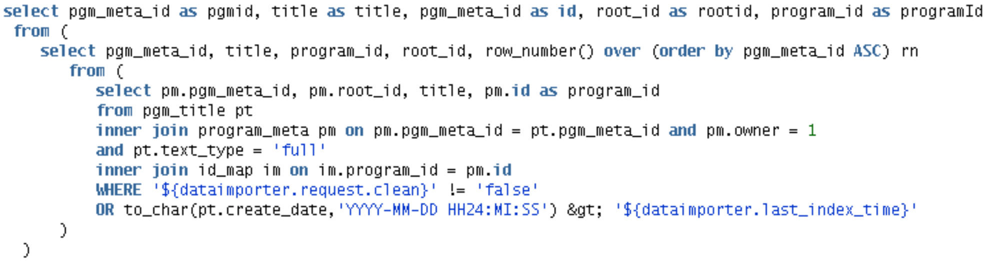
```
select venue_brand_id as venueId, venue_name as venueName from venue
WHERE '${dataimporter.request.clean}' != 'false'
OR to_char(create_date,'YYYY-MM-DD HH24:MI:SS') &gt; '${dataimporter.last_index_time}'
order by venue_brand_id
```

For Actors, it returns the program(s) where the actor has acted.



```
select rownum as paRowNum, p.id as paId, pt.title as paTitle, pm.id as paPrgId,
  (p.first_Name || CASE WHEN p.middle_Name IS NOT NULL THEN ', ' || p.middle_Name ELSE '' END || CASE WHEN p.last_Name IS NOT NULL THEN ', ' || p.last_Name ELSE '' END) as pName
from program_meta pm
inner join pgm_title pt on pt.pgm_meta_id = pm.pgm_meta_id and pt.text_size > 100
inner join pgm_credit pc on pc.pgm_meta_id = pm.pgm_meta_id
inner join person_name p on p.id = pc.name_id
WHERE '${dataimporter.request.clean}' != 'false'
OR to_char(pt.create_date,'YYYY-MM-DD HH24:MI:SS') &gt; '${dataimporter.last_index_time}'
and pm.owner = 1
```

For All Programs available

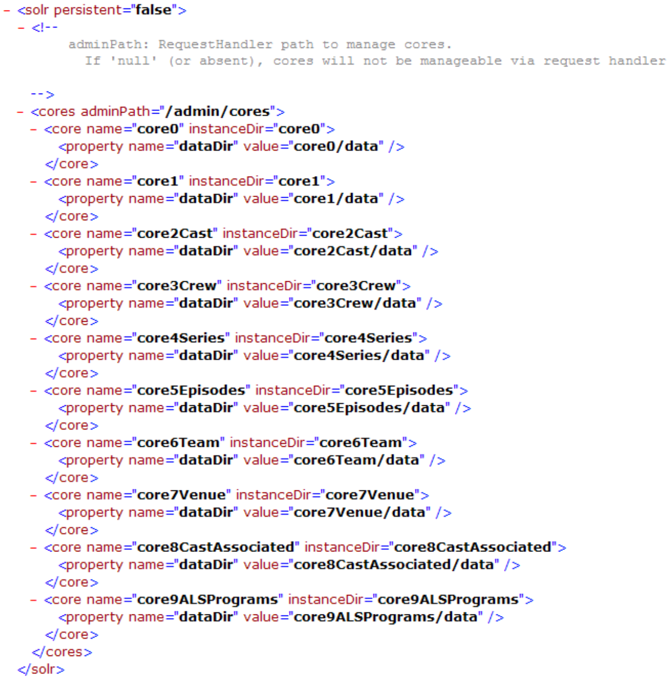


```
select pgm_meta_id as pgmid, title as title, pgm_meta_id as id, root_id as rootid, program_id as programId
 from (
    select pgm_meta_id, title, program_id, root_id, row_number() over (order by pgm_meta_id ASC) rn
       from (
          select pm.pgm_meta_id, pm.root_id, title, pm.id as program_id
          from pgm_title pt
          inner join program_meta pm on pm.pgm_meta_id = pt.pgm_meta_id and pm.owner = 1
          and pt.text_type = 'full'
          inner join id_map im on im.program_id = pm.id
          WHERE '${dataimporter.request.clean}' != 'false'
          OR to_char(pt.create_date,'YYYY-MM-DD HH24:MI:SS') &gt; '${dataimporter.last_index_time}'
    )
 )
```

2.13.2 Core Components

The following core components are created:

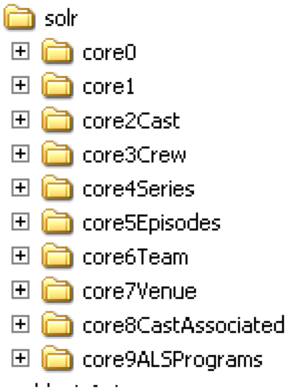
solr.xml – Defines core collection



```
- <solr persistent="false">
  - <!--
        adminPath: RequestHandler path to manage cores.
          If 'null' (or absent), cores will not be manageable via request handler

    -->
  - <cores adminPath="/admin/cores">
    - <core name="core0" instanceDir="core0">
        <property name="dataDir" value="core0/data" />
      </core>
    - <core name="core1" instanceDir="core1">
        <property name="dataDir" value="core1/data" />
      </core>
    - <core name="core2Cast" instanceDir="core2Cast">
        <property name="dataDir" value="core2Cast/data" />
      </core>
    - <core name="core3Crew" instanceDir="core3Crew">
        <property name="dataDir" value="core3Crew/data" />
      </core>
    - <core name="core4Series" instanceDir="core4Series">
        <property name="dataDir" value="core4Series/data" />
      </core>
    - <core name="core5Episodes" instanceDir="core5Episodes">
        <property name="dataDir" value="core5Episodes/data" />
      </core>
    - <core name="core6Team" instanceDir="core6Team">
        <property name="dataDir" value="core6Team/data" />
      </core>
    - <core name="core7Venue" instanceDir="core7Venue">
        <property name="dataDir" value="core7Venue/data" />
      </core>
    - <core name="core8CastAssociated" instanceDir="core8CastAssociated">
        <property name="dataDir" value="core8CastAssociated/data" />
      </core>
    - <core name="core9ALSPrograms" instanceDir="core9ALSPrograms">
        <property name="dataDir" value="core9ALSPrograms/data" />
      </core>
    </cores>
  </solr>
```

SOLR core –



```
solr
  core0
  core1
  core2Cast
  core3Crew
  core4Series
  core5Episodes
  core6Team
  core7Venue
  core8CastAssociated
  core9ALSPrograms
```

- **conf/data-config.xml** – Describes JDBC Data source, SQL Query and fields taken for indexing.
- **conf/schema.xml** – Describes field types & unique ID.
- **conf/solrconfig.xml** – Defines data-config.xml file path.

2.13.3 Request Action

The following file describes a Request Object as example:

For Cast
http://*host:port*/solr/core2Cast/select?q=castName:Alex*&rows=20&wt=json

For Crew [Any role]
http://*host:port*/solr/core3Crew/select?q=crewName:Alex*&rows=20&wt=json

For Crew [Director Role]
http://*host:port*/solr/core3Crew/select?q=crewName:A*&fq=crewRole:Director&rows=20&wt=json

For Crew [Producer Role]
http://*host:port*/solr/core3Crew/select?q=crewName:A*&fq=crewRole:Producer&rows=20&wt=json

For Series & Shows
http://*host:port*/solr/core4Series/select?q=shTitle:Bo*&rows=20&wt=json

For Episodes
http://*host:port*/solr/core5Episodes/select?q=esTitle:Frie*&rows=20&wt=json

For Teams
http://*host:port*/solr/core6Team/select?q=teamName:D*&rows=20&wt=json

For Venues
http://*host:port*/solr/core7Venue/select?q=venueName:D*&rows=20&wt=json

For Actors, it returns the program(s) where the actor has acted.
http://*host:port*/solr/core8CastAssociated/select?q=paId:12087&rows=20&wt=json

For All Programs available
http://*host:port*/solr/core9ALSPrograms/select?q=title:Do*&rows=20&wt=json

3 Issues Log

No.	Status	Date	Originator	Topic	Description
1	Open	08/04	HL	Topic1	Description1