Class OpenBISScreeningML

```
java.lang.Object
  └ OpenBISScreeningML
```

```
public class OpenBISScreeningML
extends java.lang.Object
```

Simple Matlab interface for openBIS for Screening. It is meant to be used in one Matlab session at a time, i.e. it is *not* multi-threading safe.

While written in Java, the API is idiomatic for Matlab, i.e. values are returned as multi-dimensional arrays. For the `get...` and `load...` methods the first index will contain the actual data, while the second index will contain per-row annotations. For `getFeatureMatrix`, the third index contains per-column annotations. This allows simple access with Matlab's slicing operator, see doc of e.g. `getFeatureMatrix(String)`.

A typical Matlab session looks like:

```
% Add the API jar file to the classpath
javaaddpath('/home/brinn/matlab/openbis_screening_api-batteries_included.jar')
% Login to server
OpenBISScreeningML.login('user', 'secret', 'https://www.infectome.org')

% ...perform calls on the server...

% Logout to close the session on the server
OpenBISScreeningML.logout()
```

Note: using this login your password will end up in the Matlab command history. An alternative that avoids this is to call the `Login` class. Logging in on the console will grant this class access to the openBIS server.

To learn the API one needs to understand three basic notions: code, augmented code and perm id. Space, project, experiment, plate and well have their own **code**, which is unique only in the context of the parent. That's why one needs **augmented code** to point e.g. to one experiment, because two different projects can have experiments with the same code.
Such an augmented code for experiment has a form of "/space-code/project-code/experiment-code".
For plate it has a form of "/space-code/plate-code" (note that plate code is unique on the space level).
The drawback of an augmented code is that it's not persistent. If someone e.g. moves the experiment from one space to the other augmented code of the experiment becomes invalid. That is why experiments, plates and datasets have **perm id** (permament identifier) which never change and allow to refer to them with one "magic" identifier, e.g. 20110516124520378-737166.

Author:
Bernd Rinn

Field Summary

`static java.lang.String`	**`REQUIRES_OPENBIS_AS_API`** The required version ("major.minor") of the screening API on the openBIS application server.

static java.lang.String	**REQUIRES_OPENBIS_DSS_API** The required version ("major.minor") of the screening API on the openBIS datastore server.
static java.lang.String	**VERSION** The version of the API.

Method Summary

static java.lang.Object[][][]	**getFeatureMatrix**(java.lang.String gene) Returns the feature matrix of all features for all locations (a location is one well position in one feature vector data set) connected to *gene* in [0], location annotations in [1] and feature annotation in [2].
static java.lang.Object[][][]	**getFeatureMatrix**(java.lang.String experiment, java.lang.String gene) Returns the feature matrix of all features for all locations in *experiment* (a location is one well position in one feature vector data set) connected to *gene* in [0], location annotations in [1] and feature annotation in [2].
static java.lang.Object[][][]	**getFeatureMatrix**(java.lang.String gene, java.lang.String[] features) Returns the feature matrix of the specified features for all locations (a location is one well position in one feature vector data set) in *experiment* connected to *gene* in [0], location annotations in [1] and feature annotation in [2].
static java.lang.Object[][][]	**getFeatureMatrix**(java.lang.String experiment, java.lang.String gene, java.lang.String[] features) Returns the feature matrix of the specified features for all locations in *experiment* (a location is one well position in one feature vector data set) in *experiment* connected to *gene* in [0], location annotations in [1] and feature annotation in [2].
static java.lang.Object[][][]	**getFeatureMatrixForPlate**(java.lang.String plate) Returns the feature matrix of all available features for all locations (a location is one well position in one feature vector data set) of all feature vector data sets of the given *plate* in [0], location annotations in [1] and feature annotation in [2].
static java.lang.Object[][][]	**getFeatureMatrixForPlate**(java.lang.String plate, java.lang.String[] features) Returns the feature matrix of the specified features for all locations (a location is one well position in one feature vector data set) of all feature vector data sets of the given *plate* in [0], location annotations in [1] and feature annotation in [2].
static java.lang.Object[][][]	**getGeneMappingForPlates**(java.lang.String[] platesCodes) Returns the gene mapping for the given *plateCodes* in [0] and location annotations in [1].
static java.lang.Object[][]	**getImagesMetadata**(java.lang.String augmentedPlateCode) Fetches metadata of the image datasets for the specified *plate*.
static java.lang.Object[][]	**getWellProperties**(java.lang.String augmentedPlateCode, int row, int column) Returns the properties of specified well for specified plate.
static java.lang.Object[][]	**listChannels**(java.lang.String experiment) Lists all channels measured in *experiment*.
static java.lang.Object[][][]	**listDataSetsFiles**(java.lang.String augmentedPlateCode, java.lang.String dataSetTypeCodePattern) Lists all files of all data sets for specifies plate and data set type code

	matching specified regular expression pattern.
static java.lang.Object[][]	**listExperiments**() Lists all experiment.
static java.lang.Object[][]	**listFeatures**(java.lang.String experiment) Lists all features computed for *experiment*.
static java.lang.Object[][]	**listPlates**() Lists all plates.
static java.lang.Object[][]	**listPlates**(java.lang.String experiment) Lists the plates of *experiment*.
static java.lang.Object[][]	**listSegmentationObjects**(java.lang.String plate) Lists all segmentation objects for the *plate*.
static java.lang.Object	**loadDataSetFile**(java.lang.String dataSetCode, java.lang.String pathInDataSet, java.lang.String overrideStoreRootPathOrNull) Loads file/folder of specified data set and specified file/folder path inside the data set.
static java.lang.Object[][]	**loadDataSets**(java.lang.String augmentedPlateCode, java.lang.String dataSetTypeCodePattern, java.lang.String overrideStoreRootPathOrNull) Loads data sets for specified plate code.
static java.lang.Object[][][]	**loadImages**(java.lang.String plate, int row, int col) Loads the TIFF images for the given well location, all tiles and all channels and stores them in temporary files.
static java.lang.Object[][][]	**loadImages**(java.lang.String plate, int row, int col, int tile) Loads the TIFF images for the given well location, tile number, and all channels and stores them in temporary files.
static java.lang.Object[][][]	**loadImages**(java.lang.String plate, int row, int col, int tile, java.lang.String[] channels) Loads the TIFF images for the given well location, tile number, and list of channels and stores them in temporary files.
static java.lang.Object[][][]	**loadImages**(java.lang.String plate, int row, int col, java.lang.String[] channels) Loads the TIFF images for the given well location, list of channels, and all tiles and stores them in temporary files.
static java.lang.Object[][][]	**loadSegmentationImages**(java.lang.String plate, int row, int col, int tile, java.lang.String[] objectNames) Has the same effect as loadImages(String, int, int, int, String[]), but instead of loading raw images loads their segmentation results if available.
static java.lang.Object[][][]	**loadSegmentationImages**(java.lang.String plate, int row, int col, java.lang.String[] objectNames) Has the same effect as loadImages(String, int, int, String[]), but instead of loading raw images loads their segmentation results if available.
static void	**login**(java.lang.String user, java.lang.String password, java.lang.String url) Login to the openBIS server given as *url*.
static void	**logout**() Logs out and closes the session on the server.
static void	**updateWellProperties**(java.lang.String augmentedPlateCode, int row, int column, java.lang.Object[][] properties) Updates properties of specified well for specified plate.

static java.lang.Object	**uploadDataSet**(java.lang.String augmentedPlateCode, java.lang.String dataSetFilePath, java.lang.String dataSetType, java.lang.Object[][] dataSetProperties) Uploads specified data set for specified plate.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

VERSION

```
public static final java.lang.String VERSION
```

The version of the API.

See Also:
Constant Field Values

REQUIRES_OPENBIS_AS_API

```
public static final java.lang.String REQUIRES_OPENBIS_AS_API
```

The required version ("major.minor") of the screening API on the openBIS application server.

See Also:
Constant Field Values

REQUIRES_OPENBIS_DSS_API

```
public static final java.lang.String REQUIRES_OPENBIS_DSS_API
```

The required version ("major.minor") of the screening API on the openBIS datastore server.

See Also:
Constant Field Values

Method Detail

login

```
public static void login(java.lang.String user,
                         java.lang.String password,
                         java.lang.String url)
```

Login to the openBIS server given as *url*.

Matlab example:

```
OpenBISScreeningML.login('user', 'secret', 'https://www.infectome.org')
```

Parameters:

`user` - The user id on the server
`password` - The password on the server
`url` - The URL, e.g. `https://www.infectome.org`

---

logout

```
public static void logout()
```

Logs out and closes the session on the server.

Matlab example:

```
 OpenBISScreeningML.logout()
```

---

listExperiments

```
public static java.lang.Object[][] listExperiments()
```

Lists all experiment.

Matlab example:

```
 % Get the experiments
 exps = OpenBISScreeningML.listExperiments();
 % How many experiments do we have?
 length(exps)
 % Get all information about experiment 3
 exp3 = exps(3,:)
 % Get the perm ids for all experiments
 permids = exps(:,2)
```

Returns:
Each row contains information about one experiment:

{ experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code }

---

listPlates

```
public static java.lang.Object[][] listPlates()
```

Lists all plates.

Matlab example:

```
 % Get the plates
 plates = OpenBISScreeningML.listPlates();
 % How many plates do we have?
 length(plates)
 % Get all information about plate 2
 plate2 = plates(2,:)
 % Get the simple plate codes for all plates
 codes = plates(:,4)
```

Returns:
Each row contains information about one plate:

{ plate augmented code, plate perm id, plate space code, plate code, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code }

---

listPlates

```
public static java.lang.Object[][] listPlates(java.lang.String experiment)
```

Lists the plates of *experiment*.

Matlab example:

```
% Get the plates of experiment MYEXP in project PROJ of space SPACE
plates = OpenBISScreeningML.listPlates('/SPACE/PROJ/MYEXP');
% How many plates do we have?
length(plates)
% Get all information about plate 2
plate2 = plates(2,:)
% Get the augmented plate codes for all plates
acodes = plates(:,1)
```

Parameters:
experiment - The augmented code of the experiment to list the plates for
Returns:
Each row contains information about one plate:

{ plate augmented code, plate perm id, plate space code, plate code, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code }

---

getWellProperties

```
public static java.lang.Object[][] getWellProperties(java.lang.String augmentedPlateCode,
                                                     int row,
                                                     int column)
```

Returns the properties of specified well for specified plate.

Matlab example:

```
% Get properties for well A03 of plate P005 in space SPACE
properties = OpenBISScreeningML.getWellProperties('/SPACE/P005', 1, 3)
% Get property type code of first property
properties(1,1)
% Get property value of first property
properties(1,2)
```

Parameters:
augmentedPlateCode - The augmented plate code
row - The row in the plate to get the well properties for
column - The column in the plate to get the well properties for
Returns:
A two dimensional array where the first column contains the property codes and the second column the corresponding property values.

---

updateWellProperties

```
public static void updateWellProperties(java.lang.String augmentedPlateCode,
                                        int row,
                                        int column,
                                        java.lang.Object[][] properties)
```

Updates properties of specified well for specified plate.

Matlab example:

```
% Updates properties DESCRIPTION and NUMBER for well A03 of plate P005 in space SPACE
properties = {'DESCRIPTION' 'hello example'; 'NUMBER' 3.14}
OpenBISScreeningML.updateWellProperties('/SPACE/P005', 1, 3, properties)
```

Parameters:
augmentedPlateCode - The augmented plate code
row - The row in the plate to get the well properties for
column - The column in the plate to get the well properties for
properties - A two dimensional array where the first column contains the property codes and the second column the corresponding property values.

---

getImagesMetadata

```
public static java.lang.Object[][] getImagesMetadata(java.lang.String augmentedPlateCode)
```

Fetches metadata of the image datasets for the specified *plate*.

Matlab example:

```
% Get the metadata of image datasets of plate P005 from space SPACE
imagesMetadata = OpenBISScreeningML.getImagesMetadata('/SPACE/P005');
% How many image datasets do we have? Usually there will be just one.
length(imagesMetadata)
% What is the number of tiles in the first image dataset?
imagesMetadata(1, 3)
```

Parameters:
augmentedPlateCode - The augmented plate code.
Returns:
{ images width, images height, number of tiles in the well, number of tiles rows, number of tiles columns, number of plate rows, number of plate columns } .

---

listChannels

```
public static java.lang.Object[][] listChannels(java.lang.String experiment)
```

Lists all channels measured in *experiment*.

Matlab example:

```
% Get the channels of experiment MYEXP in project PROJ of space SPACE
channels = OpenBISScreeningML.listChannels('/SPACE/PROJ/MYEXP');
% How many channels do we have?
length(channels)
% What is the name of channel 1?
channels(1)
```

Parameters:
experiment - The augmented code of the experiment to list the channels for
Returns:
Each row contains information about one channel. Currently the only information available is the channel name.

---

listFeatures

```
public static java.lang.Object[][] listFeatures(java.lang.String experiment)
```

Lists all features computed for *experiment*.

Matlab example:

```
% Get the features of experiment MYEXP in project PROJ of space SPACE
features = OpenBISScreeningML.listFeatures('/SPACE/PROJ/MYEXP');
% How many features do we have?
length(features)
% What is the name of features 1?
features(1)
```

Parameters:
experiment - The augmented code of the experiment to list the features for
Returns:
Each row contains information about one feature. Currently the only information available is the feature name.

---

loadDataSets

```
public static java.lang.Object[][] loadDataSets(java.lang.String augmentedPlateCode,
                                                 java.lang.String dataSetTypeCodePattern,
                                                 java.lang.String overrideStoreRootPathOrNull)
```

Loads data sets for specified plate code. For each data set the path to the root of the data set is returned. If it is possible the path points directly into the data set store. No data is copied. Otherwise the data is retrieved from the data store server. If the same dataset is loaded for the second time in one session it will be immediately returned from the local cache.

Matlab example:

```
% Load all data sets of plate P005 in space SPACE
dsinfo = OpenBISScreeningML.loadDataSets('/SPACE/P005', '.*', '')
% Get the data set codes
dsinfo(:,1)
% Get root path of first data set (assuming there is at least one)
dsginfo(1,2)
```

Parameters:

augmentedPlateCode - The augmented plate code.
dataSetTypeCodePattern - only datasets of the type which matches the specified pattern will be returned. To fetch all datasets specify ".*".
overrideStoreRootPathOrNull - A path, in the context of the local file system mounts, to the DSS' store root. If null, paths are returned in the context of the DSS' file system mounts.

Returns:

Each row contains information about one data set:

{ data set code, data set root path }

---

loadDataSetFile

```
public static java.lang.Object loadDataSetFile(java.lang.String dataSetCode,
                                                java.lang.String pathInDataSet,
                                                java.lang.String overrideStoreRootPathOrNull)
```

Loads file/folder of specified data set and specified file/folder path inside the data set. If it is possible the path points directly into the data set store. No data is copied. Otherwise the data is retrieved from the data store server.

Matlab example:

```
% List all data sets of plate P005 in space SPACE. The query is restricted to data sets
% of a type starting with HCS_IMAGE
files = OpenBISScreeningML.listDataSetsFiles('/SPACE/P005', 'HCS_IMAGE.*')
% Load from the first data set (assuming at least one data set found) the third file/fo
% (assuming at least three files/folders)
file = OpenBISScreeningML.loadDataSetFile(files(1,1), files(1,2,3), '')
```

Parameters:

dataSetCode - The code of the data set.
pathInDataSet - Path inside the data set pointing to the file/folder which should be down loaded. Use '/' if all files are requested.
overrideStoreRootPathOrNull - A path, in the context of the local file system mounts, to the DSS' store root. If null, paths are returned in the context of the DSS' file system mounts.

Returns:

path to the down loaded file/folder.

---

listDataSetsFiles

```
public static java.lang.Object[][][] listDataSetsFiles(java.lang.String augmentedPlateCode,
                                                        java.lang.String dataSetTypeCodePatter
```

Lists all files of all data sets for specifies plate and data set type code matching specified regular expression pattern.

Matlab example:

```
 % List all data sets of plate P005 in space SPACE. The query is restricted to data sets
 % of a type starting with HCS_IMAGE
 files = OpenBISScreeningML.listDataSetsFiles('/SPACE/P005', 'HCS_IMAGE.*')
 % Codes of all found data sets
 files(:,1)
 % Code of third data set (assuming at least three data sets found)
 files(3,1)
 % Files of third data set (assuming at least three data sets found)
 files(3,2,:)
```

Parameters:
augmentedPlateCode - The augmented plate code.
dataSetTypeCodePattern - only data sets of the type which matches the specified pattern will be returned. To fetch all data sets specify ".*".

Returns:
{data set code, file/folder paths}

---

uploadDataSet

```
public static java.lang.Object uploadDataSet(java.lang.String augmentedPlateCode,
                                              java.lang.String dataSetFilePath,
                                              java.lang.String dataSetType,
                                              java.lang.Object[][] dataSetProperties)
```

Uploads specified data set for specified plate. The data set code will be returned.

Matlab example:

```
 % Upload data set /path/to/my-data-set with properties DESCRIPTION and NUMBER for
 % plate P005 in space SPACE
 properties = {'DESCRIPTION' 'hello example'; 'NUMBER' 3.14}
 datasetcode = OpenBISScreeningML.uploadDataSet('/SPACE/P005', '/path/to/my-data-set',
```

Parameters:
augmentedPlateCode - The augmented plate code.
dataSetFilePath - Path to the data set file/folder to be uploaded.
dataSetType - Data set type.
dataSetProperties - A two dimensional array where the first column contains the property codes and the second column the corresponding property values.

---

loadImages

```
public static java.lang.Object[][][] loadImages(java.lang.String plate,
                                                 int row,
                                                 int col)
```

Loads the TIFF images for the given well location, all tiles and all channels and stores them in temporary files. The temporary files will be removed automatically when the Java Virtual Machine exits.

Matlab example:

```
 % Load the images for all channels of well B10 of plate P005 in space SPACE
 imginfo = OpenBISScreeningML.loadImages('/SPACE/P005', 2, 10)
 % Get the plate-well descriptions of all locations
 imginfo(2,:,3)
 % Show the third image (assuming there are at least three images)
 imtool(imginfo(1,3))
```

Parameters:
plate - The augmented plate code
row - The row in the plate to get the images for
col - The column in the plate to get the images for

Returns:
{ names of TIFF files, image annotation }

Each of names of TIFF files and image annotation is a vector of length of the number of images.

image annotation contains { channel name, tile number, plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code }

---

loadImages

```
public static java.lang.Object[][][] loadImages(java.lang.String plate,
                                                 int row,
                                                 int col,
                                                 int tile)
```

Loads the TIFF images for the given well location, tile number, and all channels and stores them in temporary files. The temporary files will be removed automatically when the Java Virtual Machine exits.

Matlab example:

```
% Load the images for all channels of well B10 and tile 3 of plate P005 in space SPACE
imginfo = OpenBISScreeningML.loadImages('/SPACE/P005', 2, 10, 3)
% Get the plate-well descriptions of all locations
imginfo(2,:,3)
% Show the third image (assuming there are at least three images)
imtool(imginfo(1,3))
```

Parameters:

plate - The augmented plate code
row - The row in the plate to get the images for
col - The column in the plate to get the images for
tile - The tile number. Starts with 0.

Returns:

{ names of TIFF files, image annotation }

Each of names of TIFF files and image annotation is a vector of length of the number of images.

image annotation contains { channel name, tile number, plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code }

---

loadImages

```
public static java.lang.Object[][][] loadImages(java.lang.String plate,
                                                 int row,
                                                 int col,
                                                 java.lang.String[] channels)
```

Loads the TIFF images for the given well location, list of channels, and all tiles and stores them in temporary files. The temporary files will be removed automatically when the Java Virtual Machine exits.

Matlab example:

```
% Load the images for channel DAPI of well H10 of plate P005 in space SPACE
imginfo=OpenBISScreeningML.loadImages('/SPACE/P005', 8, 10, 'DAPI')
% Get the channel names and tile numbers of all locations
imginfo(2,:,1:2)
% Show the second image (assuming there are at least two images)
imtool(imginfo(1,2))
```

Parameters:

plate - The augmented plate code
row - The row in the plate to get the images for
col - The column in the plate to get the images for
channels - The names of the channels to get the images for

Returns:

{ names of TIFF files, image annotation }

Each of names of TIFF files and image annotation is a vector of length of the number of images.

image annotation contains { channel name, tile number, plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code }

---

loadSegmentationImages

```
public static java.lang.Object[][][] loadSegmentationImages(java.lang.String plate,
                                                            int row,
                                                            int col,
                                                            java.lang.String[] objectNames)
```

Has the same effect as loadImages(String, int, int, String[]), but instead of loading raw images loads their segmentation results if available.

Parameters:
objectNames - The names of the segmentation objects to get the images for

---

loadImages

```
public static java.lang.Object[][][] loadImages(java.lang.String plate,
                                                int row,
                                                int col,
                                                int tile,
                                                java.lang.String[] channels)
```

Loads the TIFF images for the given well location, tile number, and list of channels and stores them in temporary files. The temporary files will be removed automatically when the Java Virtual Machine exits.

Matlab example:

```
% Load the images for channel DAPI of well H10 and tile 3 of plate P005 in space SPACE
imginfo=OpenBISScreeningML.loadImages('/SPACE/P005', 8, 10, 3, 'DAPI')
% Get the channel names of all locations
imginfo(2,:,1)
% Show the second image (assuming there are at least two images)
imtool(imginfo(1,2))
```

Parameters:
plate - The augmented plate code
row - The row in the plate to get the images for
col - The column in the plate to get the images for
tile - The tile number. Starts with 0.
channels - The names of the channels to get the images for

Returns:
{ names of TIFF files, image annotation }

Each of names of TIFF files and image annotation is a vector of length of the number of images.

image annotation contains { channel name, tile number, plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code }

---

loadSegmentationImages

```
public static java.lang.Object[][][] loadSegmentationImages(java.lang.String plate,
                                                            int row,
                                                            int col,
                                                            int tile,
                                                            java.lang.String[] objectNames)
```

Has the same effect as loadImages(String, int, int, int, String[]), but instead of loading raw images loads their segmentation results if available.

Parameters:
objectNames - The names of the segmentation objects to get the images for

---

listSegmentationObjects

```
public static java.lang.Object[][] listSegmentationObjects(java.lang.String plate)
```

Lists all segmentation objects for the *plate*.

Matlab example:

```
% Get the segmentation objects of plate P005 in space SPACE.
segmentationObjects = OpenBISScreeningML.listSegmentationObjects('/SPACE/P005');
% How many segmentation objects do we have?
length(segmentationObjects)
% What is the name of segmentation objects 1?
segmentationObjects(1)
```

Parameters:
plate - augmented code of the plate
Returns:
Each row contains information about one segmentation object. Currently the only information available is the segmentation object name.

---

getFeatureMatrix

```
public static java.lang.Object[][][] getFeatureMatrix(java.lang.String experiment,
                                                      java.lang.String gene)
```

Returns the feature matrix of all features for all locations in *experiment* (a location is one well position in one feature vector data set) connected to *gene* in [0], location annotations in [1] and feature annotation in [2].

Matlab example:

```
% Get feature matrix for experiment /SPACE/PROJ/MYEXP for locations connected to GENEN
fmatrix = OpenBISScreeningML.getFeatureMatrix('/SPACE/PROJ/MYEXP', 'GENENAME');
% Get the feature vector for the second location (assuming that there are at least two
% of third data set (assuming that there are at least three data sets)
fmatrix(1,:,2,3)
% Get the values of the fourth feature for all locations (assuming that there are at le
% of third data set (assuming that there are at least three data sets)
fmatrix(1,4,:,3)
% Get code of the fourth feature (assuming that there are at least 4 features)
fmatrix(3,4)
% Get the plate-well descriptions for the second location (assuming that there are at l
% of third data set (assuming that there are at least three data sets)
fmatrix(2,2,3,:)
```

Parameters:
experiment - The augmented experiment code
gene - The gene code (stored as material code in openBIS, usually it is gene id)
Returns:
a four dimensional matrix. The first dimension denotes the type in the following order: {feature matrix, annotations per location, feature codes}. The other dimensions depend on the value of the first dimension:

1. feature matrix: 2. dimension is feature vector, 3. dimension is location number, 4. dimension is data set number. If for a particular location and a particular data set the corresponding feature value does not exists NaN will be returned.
2. annotations: 2. dimension is location number, 3. dimension is data set number, 4. dimension is location annotations in the following order: {plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code, data set type}
3. feature codes: 2. dimension is feature codes in alphabetical order. 3. and 4. dimension are meaningless (i.e. they have length one)

---

getFeatureMatrix

```
public static java.lang.Object[][][][] getFeatureMatrix(java.lang.String experiment,
                                                         java.lang.String gene,
                                                         java.lang.String[] features)
```

Returns the feature matrix of the specified features for all locations in *experiment* (a location is one well position in one feature vector data set) in *experiment* connected to *gene* in [0], location annotations in [1] and feature annotation in [2].

Matlab example:

```
% Get feature matrix for features FEATURE1, FEATURE2 and FEATURE3 for
% experiment /SPACE/PROJ/MYEXP for locations connected to GENENAME
fmatrix = OpenBISScreeningML.getFeatureMatrix('/SPACE/PROJ/MYEXP', 'GENENAME', ('FEATUI
% Get the feature vector for the second location (assuming that there are at least two
% of third data set (assuming that there are at least three data sets)
fmatrix(1,:,2,3)
% Get the values of the fourth feature for all locations (assuming that there are at le
% of third data set (assuming that there are at least three data sets)
fmatrix(1,4,:,3)
% Get code of the fourth feature (assuming that there are at least 4 features)
fmatrix(3,4)
% Get the plate-well descriptions for the second location (assuming that there are at 1
% of third data set (assuming that there are at least three data sets)
fmatrix(2,2,3,:)
```

Parameters:

experiment - The augmented experiment code
gene - The gene code (stored as material code in openBIS, usually it is gene id)
features - The names of the features to contain the feature matrix

Returns:

a four dimensional matrix. The first dimension denotes the type in the following order: {feature matrix, annotations per location, feature codes}. The other dimensions depend on the value of the first dimension:

1. feature matrix: 2. dimension is feature vector, 3. dimension is location number, 4. dimension is data set number. If for a particular location and a particular data set the corresponding feature value does not exists NaN will be returned.
2. annotations: 2. dimension is location number, 3. dimension is data set number, 4. dimension is location annotations in the following order: {plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code, data set type}
3. feature codes: 2. dimension is feature codes in alphabetical order. 3. and 4. dimension are meaningless (i.e. they have length one)

---

getFeatureMatrix

```
public static java.lang.Object[][][][] getFeatureMatrix(java.lang.String gene)
```

Returns the feature matrix of all features for all locations (a location is one well position in one feature vector data set) connected to *gene* in [0], location annotations in [1] and feature annotation in [2].

Matlab example:

```
% Get feature matrix for GENENAME
fmatrix = OpenBISScreeningML.getFeatureMatrix('GENENAME');
% Get the feature vector for the second location (assuming that there are at least two
% of third data set (assuming that there are at least three data sets)
fmatrix(1,:,2,3)
% Get the values of the fourth feature for all locations (assuming that there are at le
% of third data set (assuming that there are at least three data sets)
fmatrix(1,4,:,3)
% Get code of the fourth feature (assuming that there are at least 4 features)
fmatrix(3,4)
% Get the plate-well descriptions for the second location (assuming that there are at 1
% of third data set (assuming that there are at least three data sets)
fmatrix(2,2,3,:)
```

Parameters:
gene - The gene code (stored as material code in openBIS, usually it is gene id)
Returns:
a four dimensional matrix. The first dimension denotes the type in the following order: {feature matrix, annotations per location, feature codes}. The other dimensions depend on the value of the first dimension:
1. feature matrix: 2. dimension is feature vector, 3. dimension is location number, 4. dimension is data set number. If for a particular location and a particular data set the corresponding feature value does not exists NaN will be returned.
2. annotations: 2. dimension is location number, 3. dimension is data set number, 4. dimension is location annotations in the following order: {plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code, data set type}
3. feature codes: 2. dimension is feature codes in alphabetical order. 3. and 4. dimension are meaningless (i.e. they have length one)

---

getFeatureMatrix

```
public static java.lang.Object[][][][] getFeatureMatrix(java.lang.String gene,
                                                         java.lang.String[] features)
```

Returns the feature matrix of the specified features for all locations (a location is one well position in one feature vector data set) in *experiment* connected to *gene* in [0], location annotations in [1] and feature annotation in [2].

Matlab example:

```
% Get feature matrix for features FEATURE1, FEATURE2 and FEATURE3 for GENENAME
fmatrix = OpenBISScreeningML.getFeatureMatrix('GENENAME', ('FEATURE1','FEATURE2','FEATU
% Get the feature vector for the second location (assuming that there are at least two
% of third data set (assuming that there are at least three data sets)
fmatrix(1,:,2,3)
% Get the values of the fourth feature for all locations (assuming that there are at le
% of third data set (assuming that there are at least three data sets)
fmatrix(1,4,:,3)
% Get code of the fourth feature (assuming that there are at least 4 features)
fmatrix(3,4)
% Get the plate-well descriptions for the second location (assuming that there are at l
% of third data set (assuming that there are at least three data sets)
fmatrix(2,2,3,:)
```

Parameters:
gene - The gene code (stored as material code in openBIS, usually it is gene id)
features - The names of the features to contain the feature matrix
Returns:
a four dimensional matrix. The first dimension denotes the type in the following order: {feature matrix, annotations per location, feature codes}. The other dimensions depend on the value of the first dimension:
1. feature matrix: 2. dimension is feature vector, 3. dimension is location number, 4. dimension is data set number. If for a particular location and a particular data set the corresponding feature value does not exists NaN will be returned.
2. annotations: 2. dimension is location number, 3. dimension is data set number, 4. dimension is location annotations in the following order: {plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code, data set type}
3. feature codes: 2. dimension is feature codes in alphabetical order. 3. and 4. dimension are meaningless (i.e. they have length one)

---

getFeatureMatrixForPlate

```
public static java.lang.Object[][][][] getFeatureMatrixForPlate(java.lang.String plate)
```

Returns the feature matrix of all available features for all locations (a location is one well position in one feature vector data set) of all feature vector data sets of the given *plate* in [0], location annotations in [1] and feature annotation in [2].

Matlab example:

```
% Get feature matrix for PLATECODE
fmatrix = OpenBISScreeningML.getFeatureMatrixForPlate('PLATECODE');
% Get the feature vector for the second location (assuming that there are at least two
% of third data set (assuming that there are at least three data sets)
fmatrix(1,:,2,3)
% Get the values of the fourth feature for all locations (assuming that there are at le
% of third data set (assuming that there are at least three data sets)
fmatrix(1,4,:,3)
% Get code of the fourth feature (assuming that there are at least 4 features)
fmatrix(3,4)
% Get the plate-well descriptions for the second location (assuming that there are at l
% of third data set (assuming that there are at least three data sets)
fmatrix(2,2,3,:)
```

Parameters:
plate - augmented code of the plate for which features should be loaded

Returns:
a four dimensional matrix. The first dimension denotes the type in the following order: {feature matrix, annotations per location, feature codes}. The other dimensions depend on the value of the first dimension:

1. feature matrix: 2. dimension is feature vector, 3. dimension is location number, 4. dimension is data set number. If for a particular location and a particular data set the corresponding feature value does not exists NaN will be returned.
2. annotations: 2. dimension is location number, 3. dimension is data set number, 4. dimension is location annotations in the following order: {plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code, data set type}
3. feature codes: 2. dimension is feature codes in alphabetical order. 3. and 4. dimension are meaningless (i.e. they have length one)

---

getFeatureMatrixForPlate

```
public static java.lang.Object[][][][] getFeatureMatrixForPlate(java.lang.String plate,
                                                                 java.lang.String[] features)
```

Returns the feature matrix of the specified features for all locations (a location is one well position in one feature vector data set) of all feature vector data sets of the given *plate* in [0], location annotations in [1] and feature annotation in [2].

Matlab example:

```
% Get feature matrix for features FEATURE1, FEATURE2 and FEATURE3 for PLATECODE
fmatrix = OpenBISScreeningML.getFeatureMatrixForPlate('PLATECODE', ('FEATURE1','FEATURE
% Get the feature vector for the second location (assuming that there are at least two
% of third data set (assuming that there are at least three data sets)
fmatrix(1,:,2,3)
% Get the values of the fourth feature for all locations (assuming that there are at le
% of third data set (assuming that there are at least three data sets)
fmatrix(1,4,:,3)
% Get code of the fourth feature (assuming that there are at least 4 features)
fmatrix(3,4)
% Get the plate-well descriptions for the second location (assuming that there are at l
% of third data set (assuming that there are at least three data sets)
fmatrix(2,2,3,:)
```

Parameters:
plate - augmented code of the plate for which features should be loaded
features - The codes of the features to contain the feature matrix. Unknown feature codes will be ignored.

Returns:
a four dimensional matrix. The first dimension denotes the type in the following

order: {feature matrix, annotations per location, feature codes}. The other dimensions depend on the value of the first dimension:

1. feature matrix: 2. dimension is feature vector, 3. dimension is location number, 4. dimension is data set number. If for a particular location and a particular data set the corresponding feature value does not exists NaN will be returned.
2. annotations: 2. dimension is location number, 3. dimension is data set number, 4. dimension is location annotations in the following order: {plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code, data set type}
3. feature codes: 2. dimension is feature codes in alphabetical order. 3. and 4. dimension are meaningless (i.e. they have length one)

---

getGeneMappingForPlates

```
public static java.lang.Object[][][] getGeneMappingForPlates(java.lang.String[] platesCodes)
```

Returns the gene mapping for the given *plateCodes* in [0] and location annotations in [1].

One row in the matrix corresponds to one well.

Matlab example:

```
% Get feature matrix for features FEATURE1, FEATURE2 and FEATURE for PLATECODE
genes = getGeneMappingForPlate('PLATECODE');
% Get the plate well location description of the 10th wells
loc2 = genes(2,10,1)
% Get the gene ids that are in the 10th well
geneIds = genes(1,10,:)
```

Parameters:

platesCodes - The augmented codes of the plates to get the mapping for

Returns:

{ gene ids, annotations per well } where gene ids can be 0, 1 or more gene ids. annotations per location contain:

{ plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column }
