CREATE A FEED FOR A NEW QUALIFIER TYPE

The following is an example on how to create a new qualifier_type. In this example the new qualifier_type is **SUBJ**. Refer to:

https://rolesweb.mit.edu/sys_admin_tasks.html#add_qual_type

for additional information.

1. Create the appropriate database table entries for the new qualifier_type. This example adds a new qualifier_type = SUBJ.

1.1 Create the new SUBJ qualifier_type using the following INSERT statement:

```
INSERT INTO qualifier_type
    (qualifier_type, qualifier_type_desc, is_sensitive)
   VALUES ('SUBJ', 'Subjects and Sections', 'N');
```

1.2 Run the SELECT statement below on qualifier table BEFORE choosing a base id (qualifier_id) for it to make sure you have enough room to expand. See https://rolesweb.mit.edu/sys_admin_tasks.html#add_qual_type item b for more information.

```
SELECT qualifier_type, min(qualifier_id), max(qualifier_id)
   FROM qualifier
   GROUP BY qualifier_type
   ORDER BY min(qualifier_id);
```

1.3 Run the INSERT statement below on qualifier table to create a new root level qualifier for SUBJ starting at 2000000. See https://rolesweb.mit.edu/sys_admin_tasks.html#add_qual_type item c for more information.

```
INSERT INTO qualifier
   (qualifier_id, qualifier_code, qualifier_name,qualifier_type, has_child,
      qualifier_level, custom_hierarchy)
   VALUES (2000000,'ALL_SUBJECTS','All academic courses and
   subjects','SUBJ','N',1,'N');
```

1.4 Run the SELECT statement below to get the value to use for the qualifier_id for the INSERT statement in 1.5. See https://rolesweb.mit.edu/sys_admin_tasks.html#add_qual_type item d for more information.

```
SELECT max(qualifier_id)+1 FROM qualifier WHERE qualifier_type = 'QTYP';
```

The qualifier_id for the SUBJ qualifier_type will be 120038.

1.5 Run the INSERT statement below to create a new qualifier (for the SUBJ qualifier_type) in the qualifier table. See https://rolesweb.mit.edu/sys_admin_tasks.html#add_qual_type item e for more information.

```
INSERT INTO qualifier
   (qualifier_id, qualifier_code, qualifier_name, qualifier_type, has_child,
      qualifier_level, custom_hierarchy)
   VALUES (120038, 'QUAL_SUBJ', 'Qualifier type: SUBJ', 'QTYP', 'N', 2, 'N');
```

1.6 Run the SELECT statement below to find the root-level qualifier for qualifier_type QTYP.

```
SELECT qualifier_id from qualifier
   WHERE qualifier_type = 'QTYP'
   AND qualifier_level = 1;
```

The qualifier_id for the qualifier_type QTYP will be 120000.

1.7 Use the qualifier_id of the root-level qualifier QTPY (120000) and the qualifier_id of the new qualifier type SUBJ (120038) and runt the INSERT statement below to create a record in the QUALIFIER_CHILD table linking parent and child.

```
INSERT INTO qualifier_child (parent_id, child_id)
   VALUES (120000, 120038);
```

1.8 Run the INSERT statement below to create a similar record in the QUALIFIER_DESCENDENT table.

```
INSERT INTO qualifier_descendent (parent_id, child_id)
   VALUES (120000, 120038);
```

1.9 Run the INSERT statement below to create a record in the rdb_t_roles_parameters table to create the maximum number of actions permitted in a single transaction when processed by a feed.

```
INSERT INTO rdb_t_roles_parameters
   (parameter,value,description,default_value,is_number,update_user,update_ti
      mestamp)
   VALUES ('MAX_SUBJ',2000,'Maximum number of actions allowed for SUBJ
      qualifiers',1000,'Y','BSKINNER',NOW());
```

2. Create a new pm file for the feed. This example adds a feed for qualifier_type = SUBJ

2.1 cd to /perMIT/feeds/roles_feed. Make a copy roles_org2.pm and rename the copied file to roles_subj.pm

2.2 Edit roles_subj.pm and find all occurrences of **org2** and replace with **subj** (this is case sensitive so be certain that you preserve the case).

2.3 find the line that contains:

```
'SUBJ', '10000000', '', 'MIT-All';
```

and change to

```
'SUBJ', 'ALL_SUBJECTS', '', 'ALL academic courses and subjects';
```

2.4 find the select statement that begins with:

```
$stmt = "select o.node_id, o.parent_node_id, substr(o.title, 1, 50),"
```

and change the entire select statement to:

```
$stmt = "SELECT DISTINCT 'SCHOOL_'||school_code,'ALL_SUBJECTS',school_name
   FROM WAREUSER.whsis_department
   UNION
      SELECT sis_department_code,'SCHOOL_'||school_code,sis_department_name
      FROM WAREUSER.whsis_department
   UNION
```

```
   SELECT master_subject_id || '-' || substr(term_code,3,4),
      master_course_number, subject_title || ' (' ||master_subject_id||
      ')'
   FROM WAREUSER.whsubject_offered
      WHERE term_code >= '2011FA'
      AND master_subject_id = subject_id
      AND is_not_section = 'Y'
UNION
   SELECT 'SCHOOL_SP','ALL_SUBJECTS', 'Quasi-school of  Special Programs'
   FROM dual
UNION
   SELECT 'SP', 'SCHOOL_SP', 'Special Programs'
   FROM dual
ORDER BY 1";
```

Below are the two SQL statements that were provided from the DATA WAREHOUSE team, the two statements were combined into the single statement used above. Use sqlplus to develop the select statement.

A. To get a full list of hierarchy members:

```
SELECT school_code, school_name FROM WAREUSER.whsis_department
UNION
SELECT sis_department_code, sis_department_name FROM
WAREUSER.whsis_department
UNION
SELECT master_subject_id||'-'||substr(term_code,3,4), subject_title
FROM WAREUSER.whsubject_offered
   WHERE term_code IN ('2011FA','2011SP')
   AND master_subject_id = subject_id
   AND is_not_section = 'Y';
```

B. To get the parent-child relationships:

```
SELECT 'MIT_SUBJECTS', school_code FROM WAREUSER.whsis_department
UNION
SELECT school_code, sis_department_code FROM WAREUSER.whsis_department
UNION
SELECT master_course_number, master_subject_id||'-'||substr(term_code,3,4)
FROM WAREUSER.whsubject_offered
   WHERE is_not_section = 'Y' AND master_subject_id = subject_id AND
        term_code IN ('2011FA','2011SP')
```

2.5 Save the above changes.

3. cd to /perMIT/feeds. Edit roles_feed.pl

3.1 At the end of the MODIFICATION HISTORY section, add the following line:

```
# 6/1/2010 DSPS - Add SUBJ processing
```

Find the sub **ProcessArguments** sub-routine, and before the **#not needed** comment, add the following line:

```
require roles_subj; # For Academic courses and subjects
```

Save the above changes

4. cd to /perMIT/feeds. Edit run_procedures.pl

4.1 At the end of @proc_name, change from:

```
'Roles: Extract Department information from MDH ',
'Roles: Compare Department Information from MDH and Roles DB tables',
'Roles: Load Department Information changes to Roles DB tables');
```

To:

```
'Roles: Extract Department information from MDH ',
'Roles: Compare Department Information from MDH and Roles DB tables',
'Roles: Load Department Information changes to Roles DB tables',

'Roles: Extract academic subject information from Warehouse ',
'Roles: Compare academic subject information from Warehouse and Roles DB
   tables',
'Roles: Load academic subject changes to Roles DB tables');
```

4.2 At the end of @proc_cmd, change from:

```
$dir . 'roles_feed.pl roles_extract dept mdept',
$dir . 'roles_feed.pl roles_prepare dept roles',
$dir . 'roles_feed.pl roles_load dept roles');
```

To:

```
$dir . 'roles_feed.pl roles_extract dept mdept',
$dir . 'roles_feed.pl roles_prepare dept roles',
$dir . 'roles_feed.pl roles_load dept roles',

$dir . 'roles_feed.pl roles_extract subj warehouse',
$dir . 'roles_feed.pl roles_prepare subj roles',
$dir . 'roles_feed.pl roles_load subj roles');
```

Save the above changes and you are done.