P6 Extended Schema

Advanced Techniques

Contents

Top Five Things You Need to Know about the P6 Extended Schema…………….. 2
Do NOT Turn on Publication or Global Services Until You are Prepared………. 2
Plan Ahead for DB Log File Growth……………...................................……….2
Turning on Global Services will NOT Publish Your Projects……………………...2
The Arbiter Controls EVERYTHING Project Related…………………………….3
Your Transactional DB will Grow……………………………………………………. 3
Database……………………………………………………………………………….. 4
Log Growth…………………………………………………………………………….. 4
Transactional DB Growth……………………………………………………………..5
Starting Over………………………………………………………………………… 6
Restoring the Database……………………………………………………………… 6
Defining Date Range…………………………………………………………………..9
Turning on Publication………………………………………………………………...10
Publishing Too Often…………………………………………………………………..11
Trigger Update…………………………………………………………………………11
P6 8.2 Service Pack 2………………………………………………………………11
Only One App Server…………………………………………………………………11
LOGGING………………………………………………………………………………12
Information on Job Failures…………………………………………………………. 12
How to Clean Up JOBSVC Table Automatically……………………………………12
Admin App Settings……………………………………………………………………. 13
More Threads to Publish More Projects……………………………………………..13
For Large Data Sets or If Seeing Timeouts……………………………………… 13
Dependencies…………………………………………………………………………..14
When to Run Which Service…………………………………………………………14
Projects Enabled by Default…………………………………………………………. 14
Baseline Data…………………………………………………………………………... 15
Baseline Views and Data……………………………………………………………... 15
Tips and Tricks…………………………………………………………………………. 16
I Want to Build My Own Queue……………………………………………………… 16

Top Five Things You Need to Know about the P6 Extended Schema

If you have only a few minutes and need a high-level reference for the most important things to know about the P6 Extended schema, here they are:

Do NOT Turn on Publication or Global Services Until You are Prepared

Just turning on services to 'see how they work' is not the best approach. If you want to test the system, it is HIGHLY recommended that you proceed slowly. Begin with all projects off and turn on only a few at a time to see what the Extended schema has to offer. Opt In rather than all on by default. If you also use P6 Analytics, this point is even more important because the P6 Extended Schema is the life line for P6 Analytics data. Everything from projects to codes to spreads and date ranges of the data warehouse depend on Extended Schema settings.

Plan Ahead for DB Log File Growth

Depending on how often you publish your projects and the number of changes each publication entails, having *archivelog* mode enabled can result in a substantial amount of disk space being expended on archive logs from extended schema transactions. This is normal. This occurs due to the extended schema making a large number of inserts based on changes.

It is important to plan ahead and have plenty of disk space available. Have alerts set up to notify you of growth. Have a plan on how to clean up the logs and on how long you need to keep them. In reference to ‘Do NOT Turn on Publication or Global Services Until You are Prepared’, turning on the extended schema for all projects is not what you want to do, especially for larger customers.

Before turning the system on, have a plan for turning on groups of projects and work to balance this with your database server. The last thing you want is to turn the system on for all projects and accumulate a massive volume of archive logs that deplete the disk space on your database server. Obviously, this is a worst case scenario and rarely occurs, but it is something to plan and be prepared for. It cannot be repeated often enough: Plan ahead!

Turning on Global Services will NOT Publish Your Projects

Consider global and project services as completely different entities. You need a plan for each. When does it make the most sense for your organization to run each type of service? For global, running once a day might be fine. For project services, you need your arbiter settings (thresholds for changes and time) to match what works best for your organizational reporting needs. Planning your global and project services schedules is a detailed subject; the main point to understand here is that running Global Services does not publish any project data. See P6 Administration guide for more information on Global and Project services.

The Arbiter Controls EVERYTHING Project Related

The arbiter controls exactly which projects get published and when. It is good to understand how the arbiter determines this, so that if your projects are not publishing as expected, you will have a better idea of how to rectify the problem.

Your Transactional DB will Grow

The data that is calculated by the Extended schema services is held in the ‘X tables.’ X tables is another way of referring to tables like ProjectX which is the Extended Schema partner for Project table. The X tables do not store history. It is not unreasonable to expect your transactional database could grow by 25-50% depending on the number of projects published and customer data size. There are charts in the P6 Extended Schema white paper and P6 Administration guide discussing database growth. Again, plan ahead.

Database

The Extended schema’s data resides in the P6 transactional database. For this reason there are some areas of the database you should be aware of and before turning on the project and global services.

Log Growth

With the Extended schema, Archive logs can grow quickly on the database server because of the large number of inserts generated by all the running services. As changes occur, rows are removed and inserted. The archives generated are directly proportional to the amount of change happening in the database. During project publishing, changes can be due to update, insert or delete statements on the DB. When multiple projects are published at one time, substantial archive log growth is not unusual.

The key to controlling growth is determining how often you need to update your projects based on the frequency of changes. By default, projects are published when they either reach 100 changes or have not been published within a specified time frame (for example, 8 hours) and have at least one change. Projects can also be published manually by selecting ‘Publish Now’ through the P6 application. The ideal situation is to have the system publish small numbers of projects on a continual basis. You do NOT want to set the number of changes threshold to something extremely low because that could cause projects to be published continuously and flood the system. You also do not want to specify too long of an interval between publications because that could cause the arbiter to publish all projects in one massive group. The default values have been tested and found to provide a reasonable balance between the number and frequency of projects published.

You may need to adjust the default values based on your system and how often you see projects being published. This is a key reason for turning on groups of projects. Opt In only the projects you need immediately; then opt in other projects as requirements demand. This will allow you to monitor log growth, db usage, and growth and to adjust update frequency as needed.

The most important thing to be aware of is that these log files will grow. Extended Schema services run continuously throughout the day. Based on the number of projects being published, the number of changes in each, and their publishing thresholds, you should be able to recognize spikes in database usage during the day. Tracking these times and preparing your database server with an adequate amount of disk space is key. The DBA should design a proper backup schedule on the database server for transferring archives at regular intervals to ensure that adequate disk space is available for regular usage and to avoid any possible system down time.

Transactional DB Growth

P6 extended schema is written to the 'X tables’ (as described above) in the transactional schema. These 'X' tables are so labeled because they are the traditional transactional table name (TASK) with an X added to the end (TASKX). These X tables hold calculated values that are not normally stored in the transactional database. There are other tables that do not have a transactional counterpart, such as ActivitySpread, ReportDate, and WBSHierarchy to name a few. To avoid storing duplicate information, the X tables are comprised of object IDs and the calculated values associated with them. Based on how many projects you are publishing and the size of your current transactional database, you can expect some growth as the publication and global services are run for the first time. Generally these tables will then grow only as much as your transactional database grows. The X tables do not store historical values, so they will not grow continuously but only as the PMDB grows.

Deleted projects and activities, which do not show when running reports against the extended schema views, remain in the X tables until cleaned up by the CLEAN_PX_DELETE procedure (For more information on the CLEAN_PX_DELETE procedure see P6 Administration guide). This clean up procedure should not be run too often – perhaps monthly, during downtime or upgrades. Generally, the deleted records do not have a large effect on transactional database size or performance.

Starting Over

In certain situations customers may find they want to start over again populating their extended schema (maybe they accidently turned it on before they were ready). This can be done by deleting or truncating all the data in the X and related Extended schema tables as well as removing some entries from the SETTINGS, JOBSVC, and JOBLOG tables. Doing this can restore the database to its initial state so that Publication can be turned on with proper planning in place. The following sections will walk you through this process.

Restoring the Database

We will begin by cleaning up the X tables. Included in Step 1 (below is the SQL) we will turn off publication for all projects and then remove the data from the X tables.

Step 2 of the process we will clean up the 'Px' settings which contain dates for the last time these values were updated. If these rows are not removed, they could impede updates of extended schema data in the future. The JOBSVC table should also be cleaned of this data.

Step 1: Wipe out Extended Schema data. Truncate or delete depending on data size.

```
CONNECT as ADMUSER (for Oracle; for SQL Server, connect to the p6 db)

---TURN OFF FOR ALL PROJECTS

update project set px_enable_publication_flag = 'N';

commit;

---DELETE FROM ALL X TABLES

delete from activitycodetypesecurity;

delete from activitycodehierarchy;

delete from activityspread;

delete from actvcodex;

delete from budgchngx;

delete from calendarx;

delete from costaccounthierarchy;

delete from costsecurity;

delete from currtypex;

delete from documentx;

delete from epshierarchy;

delete from epsspread;

delete from pcatuserx;
delete from pcatvalx;
delete from pfoliox;
delete from plprojref;
delete from profprivx;
delete from projcostx;
delete from projectspread;
delete from projectsecurity;
delete from projectcodehierarchy;
delete from projectx;
delete from projfundx;
delete from projissux;
delete from projpcatx;
delete from projwbsx;
delete from rcatvalx;
delete from reportdate;
delete from reporttime;
delete from resourceassignmentspread;
delete from resourcecodehierarchy;
delete from resourcehierarchy;
delete from resourcelimit;
delete from rfoliox;
delete from riskrspplnx;
delete from riskx;
delete from rlfoliox;
delete from roleratex;
delete from rsrccurvx;
delete from rsrchourx;
delete from rsrcratex;
delete from rsrcrcatx;
delete from rsrcrolex;
delete from rsrcsecx;
delete from rsrcx;
delete from taskactvx;
delete from taskdocx;
delete from taskfinx;
delete from taskmemox;
delete from taskpredx;
delete from taskprocx;
delete from taskriskx;
delete from taskrsrcx;
delete from taskx;
delete from timeshtx;
delete from trsrcfinx;
delete from udfcodex;
delete from udfvaluex;
delete from userobsx;
delete from usersx;
delete from wbsbudgx;
delete from wbsmemox;
delete from wbsrsrc_qtyx;
delete from wbsrsrcx;
delete from wbsspread;
delete from wbsstepx;
delete from wrk_log_results;
commit;
```

Step 2: Clear out any px settings and timestamps

```
delete from settings where namespace like 'Px%';
delete from settings where namespace like 'PX%';
commit;
---Clear out all jobs
delete from joblog where job_id in (select job_id from
jobsvc where job_type in
('JT_ProjectArbiter','JT_EnterpriseData','JT_Security','JT_EnterpriseSum','JT_ResourceMgmt','JT_Project'));
```

```
delete from jobsvc where job_type in
('JT_ProjectArbiter','JT_EnterpriseData','JT_Security','JT_EnterpriseSum','JT_ResourceMgmt','JT_Project');

commit;

--------------------------

---How to clean up PL_ProjRef table (orphans)

--------------------------

DELETE from plprojref where proj_id not in (select proj_id from project);

---this can be used to delete orphan records for deleted projects even after publication has been initiated

commit;
```

Defining Date Range

Before turning on Publication, you should define your date range in the P6 application. Setting up a date range that is too long can have performance issues, especially in P6 Analytics. Your date range setting will cover how much spread data is created for every project. For example, if you pick today and 2 years into the future, you will get spread records for every day of this date range. If you set the date range for 20 years, you will get spread records for every day for every project for 20 years. This can add up quickly. The date range is a rolling value, so setting just a few years in the future and in the past will generally be sufficient. Any additional data outside of the date range will still be included; it will just not have a daily spread record for it. But if a project has remaining units and remaining costs 15 years in the future, those values will still be counted in the totals even if the date range only covers 2 years.

Plan properly for your date range. If you change it in the application, it will force a republication of all projects and basically start the process over for all data. This should be avoided.

In P6 Analytics, the same date range is used, as in the extended schema, to determine spread data. If you define an extremely large period, all that spread data must be extracted which will add to the ETL run time and could degrade performance.

The data range can also be inserted manually into the database if so desired. It is recommended that the date range be entered through the application. A warning, changing the date range in the application will cause an automatic full recalculation of everything. For this reason it is sometimes an easier process to handle through SQL. Here is an example of how to do this:

```
CONNECT as ADMUSER (for Oracle, for SQL Server connect to the p6 db)


---INSERT DATE RANGE

insert into SETTINGS (namespace, setting_name, setting_value)

values ('PxService.DateRange','StartDate','2009-01-01 00:00:00');

insert into SETTINGS (namespace, setting_name, setting_value)

values ('PxService.DateRange','RollingInterval','YEAR');
```

```
insert into SETTINGS (namespace, setting_name, setting_value)

values ('PxService.DateRange','RollingCount','3');
```

Turning on Publication

You can turn on publication project through the P6 application, or you can do it from the database. You can define additional WHERE clause criteria to turn on only those projects that you want. Any criteria can be used: code assignments, EPS, etc.

Here is an example of how to turn on publication for all projects:

```
CONNECT as ADMUSER (for Oracle; for SQL Server connect to the p6 db)

---TURN ON FOR ALL PROJECTS

update project set px_last_update_date= null, px_next_date = sysdate, px_safety_date = null,
px_enable_publication_flag = 'Y', px_priority = 50;

---safety_date is key for existing projects to make sure they all get published

---next_date being set to sysdate makes it publish on the next run of the arbiter
```

These changes will cause all projects to be picked up the next time the arbiter runs. Again just to revisit control, it is highly recommended to opt in only a few projects or a filtered group of projects for the first run. In the SQL example above you would want to add a conditional statement so as to not opt in all projects.

Publishing Too Often

At times, you might feel that projects are being published too often. Perhaps there were no changes and a project was published. Or you just scheduled and the project was published right away. There are several reasons why this may be happening.

Trigger Update

If you schedule, apply actuals, level, or summarize a project in P6, it can update one of these five fields:

- STEP_COMPLETE_FLAG
- CLNDR_ID
- SUM_BASE_PROJ_ID
- DEF_COST_PER_QTY
- LAST_RECALC_DATE

A change to any of these fields forces a sysdate timestamp on px_next_date on the row in the PROJECT table. This means that the next time the arbiter runs, the project will be published. To prevent that, disable the trigger TR_PX_PROJECT.

P6 8.2 Service Pack 2

Before P6 8.2 SP2, there was an issue where projects could be published after having crossed the time threshold but without having any changes. To avoid this, ensure that you are using P6 8.2 SP2 or later. The issue is fixed in P6 8.2 SP2 and later to require the time threshold and at least one change for a project to be published.

Only One App Server

If there is another app server running against the P6 PMDB, it can begin picking up publication projects and running global services. Ensure that the app server you are working on is not adding new jobs and that only the number of threads allocated for project publication appear in the JOBSVC table. For example, if you specified four threads for project publication in the Admin App, you should never see more than four projects running at the same time in JOBSVC.

LOGGING

Information on Job Failures

In the database, you can view the JOBLOG table and view the BLOB field as text. The following command will display all the log rows for Project service jobs:

```
select * from joblog where job_id in (select job_id from jobsvc where job_type = 'JT_Project')
```

You can also view the physical logs on the application server by navigating to the P6 home directory and accessing \p6\WebAccessLogs\services\jt_project. This will display a log for each project service. The \services folder also contains a section for each of the global services as well, so you can investigate the details of any job failures there.

How to Clean Up JOBSVC Table Automatically

The setting to clear out Completed ASAP jobs from the JOBSVC table is located in the P6 admin app under Configurations\Custom\Services\ASAP Cleanup Rate (1d).

When this threshold passes, completed ASAP jobs like JT_Project should get cleared out. This is necessary to prevent the number of rows in the table from becoming excessive after a project is published a few times.

Admin App Settings

More Threads to Publish More Projects

In the P6 Admin app, you can define additional threads for the publication of projects. Two threads are allocated by default, but if your system has the necessary resources, this can be increased. Four to eight threads have worked well with some of our larger customers.

The setting is located in the Admin application at Custom\Configuration Name\Services\Publication\Services\Publish Project -- Concurrent Threads. (Path as of EPPM 8.3 release)

For Large Data Sets or If Seeing Timeouts

The default duration for allowing queries to run can be too small for larger data sets. If you see failures and timeouts you will want to increase these values.

Configurations\Custom\Configuration Name\Database\Instance\Connection Pool (Long Running)

-MaximumLeaseDuration - default 15m

-this can be increased to 1h.

Configurations\Custom\Configuration Name\ThreadPool

-MaxLongRunningTaskDuration

-this can be increased to 1h.

-MaxTaskDuration

-this can be increased to 20m.

Configurations\Custom\Configuration Name\Database\Instance\Connection Pool (Regular)

-MaximumLeaseDuration - default 2m

-this can be increased to 20m.

Dependencies

When to Run Which Service

In what order should global services (Enterprise Data, Enterprise Summaries, Security, Resource Management) and project publication (which handles each project and project related data) be run?

Run all your global services first, and then run your project publication. Then, rerun Enterprise Summaries (this will perform aggregations and roll up summary data from the activity to the EPS level). The result will be that all projects are published and are available under the EPS bands.

Next, run the Security service. Any changes that have been made to OBS access, project privileges, global profiles, resource access, adding a new project, etc. require the security service to be rerun. If you are doing reporting later in the day or the next day it is probably not as important to run right away. You can let the natural progression and execution of this global service occur. This service should be set to run daily to make sure to capture all security changes. This way you don't have to worry about manually running the service, let the service schedule handle this for you. If you are using P6 Analytics, however, you will need to run this service if any of the aforementioned changes have occurred. P6 Analytics will look for security changes; if a user requires access to a project, the project must have security calculated for it.

When security is calculated, it writes rows to the ProjectSecurity table in PMDB. This contains a row for each project and each specific user that has access to it. If you have a new project but you cannot see it through the reporting views or P6 Analytics, more than likely it is because the security service needs to be run. The data is most likely there and calculated, you just don't have access.

Projects Enabled by Default

Following a P6 database upgrade, all projects will have the px_enable_publication_flag set to 'Y' by default, therefore enabling them all to be published. This condition is for databases being upgraded from a pre-extended schema release. If coming from a version where the extended schema already existed the setting will be respected. Also copying or adding new projects can be enabled by default also. If adding through the P6 web application there is a setting in the p6.ear and will require a change in a future release to set the default to'N'. This is an enhancement request. If you are using the P6 client application and add projects this way, the default value can be turned off. The client reads the value from the database. It can be set to 'N' for

```
CONNECT as ADMUSER (for Oracle, for SQL Server connect to the p6 db)

----------------------------------------

---How to change the default enable publication value for a project

----------------------------------------

ALTER TABLE PROJECT modify px_enable_publication_flag DEFAULT 'N';
```

Baseline Data

Baseline Views and Data

Baseline data was added to the P6 Extended Schema after the 8.1 release. Functionality is available in the p6 application to enable publishing baselines. Check the *Enable Baseline Publication* check box at Administer\Application Settings\Services.

From here you can go to each project and view the project baselines. There will be a check box next to each baseline so you can choose which baselines to publish.

Getting the baselines to publish can sometimes be a little tricky because they are not automatically queued up. One way to accomplish this is through the database by setting the px_next_date = sysdate for these baseline project rows in the PROJECT table. Or in the application, restore and read as a baseline or make a change that would affect the baseline.

After they are published once, baselines generally will not need to be published again unless a change is made, and at that time when a change is made the project will be queued up like normal projects are based on the threshold settings.

There are new views in Pxrptuser – BLActivity, BLProject, BLWBS – which can be joined on to access baseline project data. Baseline project data is integrated into the other existing views. Joining to either the Project or BLProject view will determine if you are getting baseline projects or just normal projects.

Tips and Tricks

I Want to Build My Own Queue

One request that has come up before was having an arbiter for specific groups of projects so those changes are only pulled over at a designated time. For example, I have a bunch of projects under one EPS that should only be reported on once a week. I don't want anyone to be able to report on those changes until that time. Through Px, the only real way out of box is to disable publication and then turn it back on.

If you are using P6 Analytics or the P6 Reporting Database, there are filtering capabilities that would allow you to have an ETL for each group. One way for Px would be to write a custom automation to insert these rows directly into the JOBSVC table. The projects would never be marked as *Enable Publication*; we would just force the records into the DB and the job service frame work would pick them up.

The goal of this section is to discuss how a group of projects can be separated from the ‘global’ Px project service execution. The arbiter service has global settings across all projects that have enable publication set to ‘Y.’

What if I had a group of projects that I wanted to run on a delay? This group of projects needs only be published once a month. And because of reporting purposes I don’t want it to be updated anymore. First thing is to determine this subset of projects and get the project ID. Preferably you can determine whether these projects are located within a certain EPS, or contain a specific activity code, etc.

I would suggest using PL\SQL to populate a new table you would create in your admuser schema called pxprojectlist. In this pxprojectlist, populate a column for proj_id of these projects. Now that you have a list of the project IDs you want to be members of this sub group, turn off enable publication on these projects to make sure the arbiter does not detect them. For each project, make sure the px_enable_publication_flag = ‘N’ for these rows in the Project table.

Now lets look at how inserting a job manually would look. Below is the syntax for inserting a job for a specific project ID:

```
  INSERT INTO jobsvc (job_id, seq_num, audit_flag, job_name, job_type, job_type_key, recur_type,
status_code, submitted_date)

  VALUES (1111, 50 , 'N','PX Project ' || 4420, 'JT_Project', 4420, 'RT_WebASAP', 'JS_Pending', sysdate);

---in values the Px Project is just text and can be changed.  4420 will be the project ID, replace this with the
project ID of the service you would like to run.

Commit;
```

If you update the project ID with one of your projects and insert this into the Jobsvc table, the project job will be picked up and executed. It is 100% independent of the arbiter. You don’t need to add an additional temporary arbiter or even need to actually have an arbiter at all. This is its own job service and has no relation to the arbiter.

Now that you have a list of project IDs and you know how to insert a job service record for a specific project, the next step is to write PL\SQL to populate a loop of project IDs and create as many insert statements as necessary. This will allow you to avoid having to use hard code project IDs and to have the records inserted for any project that meets the pre-determined criteria.

]Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com