Running Multiple Projects

Technical Article

Running multiple projects side by side in OL Connect

Connect Workflow can only run a single Workflow configuration at a time. This means that if you have multiple solutions or projects that need to run on the same server, you need a way to integrate those into a single Workflow configuration before deploying. For the online samples that OL are hosting (Accounts Receivable and Proof of Delivery), we have solved this by standardizing the way these solutions are created in a way that allows them to be combined easily. This still involves a manual process, but, when done right, it allows a straight forward deployment without having to change the actual solutions.

This info applies to both PlanetPress Connect Workflow, as well as to PReS Connect Workflow, so we will either refer to it as OL Connect Workflow, or an abbreviation of that.

Standardize projects

Every solution or project should be organized in the same way. Usually, there are two categories of files: configuration files that don't change at run time, and run time files that do change. We recommend having a strict separation between these. Put all static configuration files in a folder structure, and have a separate folder structure for all run time files. Ours look like this:

- \OL Connect Solutions\
    - Common\
    - AR\
    - POD\
    - Integration\
- \OL Connect Workspace\
    - Common\
    - AR\
    - POD\

On the configuration side, we also have a standardized structure:

- \OL Connect Solutions\
    - <project name>\
        - Configurations\

- `Connect resources\`
- `Workflow\`
    - `Workflow-scripting\` (optional, any externally saved scripts go here)

If a project has other configurations beyond the OL Connect resources and Workflow config (for instance, we use WordPress when a web front-end is needed), then these will sit in separate folders in the Configurations\ folder. Static resources that are configurations sit one level higher (sample data for instance).

Special projects

Common

To allow sharing of resources, any configurations or resources that are used by multiple projects, and that do not need changing to be used, are put in the "`OL Connect Solutions\Common`" folder. This can include scripts, OL Connect resources, etc.

Integration

Anything that needs manual integration to allow multiple projects to co-exist, is put in the "`OL Connect Solutions\Integration`" folder. The main thing that can be found here, is the combined Workflow config with all projects in it, and perhaps the starting point for that integrated config, for instance if a common startup process is needed, or if there are global variables that have a scope beyond a single project.

If the individual projects need those "global globals" also to be able to run (try to avoid this if possible), then these variables can exist in the individual project's Workflow configs as well.

Auto deploy

To keep the distinction between static resources and run time data strict: anything for which a static resource is the starting point, but that can then change at run time, needs to be deployed automatically from the solutions folder into the workspace at startup. (A startup process could check for the file to exist in the workspace, if it's missing, it will copy the static file from the solutions folder to the workspace.)

Externalize environment specific parameters

Anything that is specific to a DTAP environment, should be in an external to Workflow configurations, OL Connect resources, etc. This allows promoting configurations through DTAP without having to change them in between (which would partially defeat the purpose of having DTAP). For our solution samples, we use a JSON file that has an object with settings for each DTAP environment:

```
{
    "systems": [{
        "description": "Production server",
        "hostname": "PROD1",
        "publicURL": "https://prod.sample.com/solutions",
        "frontendServer": "https://acme-prod.azurewebsites.net"
    },
    {
        "description": "Test server",
        "hostname": "TEST1",
        "publicURL": "https://test.sample.com/solutions",
        "frontendServer": "https://acme-test.azurewebsites.net"
    },
    {
        "description": "Dev server",
        "hostname": "DEV3",
        "publicURL": "https://dev.sample.com/solutions",
        "frontendServer": "https://acme-dev.azurewebsites.net"
    }]
}
```

At At startup, this file is read and the right settings are selected based on the host name. We can have as many different environments as we like with this setup.

Project specific guidelines for Workflow

Namespace prefix for project specific globals

Anything that is global to a Workflow config, needs to get a project specific prefix, to avoid clashes when projects are combined. It helps if that prefix is short. For instance, we choose “ar” and “pod” for our samples. This does not only apply to Global Variables, but also to process names, the “action” in Http Server Input tasks, subprocess names, OL Connect resources such as templates, data mappings, etc.

Use grouping

Processes, subprocesses, and global variables should always be in groups (it helps to use the name space prefix for the top level group). Global variables and subprocesses only have one grouping level, so when you have many of those, you can have multiple groups per project, and then you should use the name space prefix for each group.

Starting with OL Connect 2018.2, multiple startup processes are allowed, and process groups can be nested (groups within groups). This means that every project can have its own startup process (if it needs one), and it’s possible to create groups of processes within a project.

Common global variables and (sub)processes

If there are any processes or variables that have a broader scope than a single project, these should go into a separate Workflow configuration. This config should go into a special “integration project”.

Changing a project

Whenever a project needs some kind of change, the change is done on the project's Workflow config, and that config is then modified and tested in isolation. Once the changes are done, and everything seems to work in the development environment, there are two options:

1. Deploy just that project in the test environment and have it tested in isolation. This is preferred if there is no continuous testing going on with all projects, and if there is a separate test team that doesn't have access to the development environment. To deploy:
2. Just copy "`\OL Connect Solutions\project X\`", over to the test server, and send that project's Workflow to the local Workflow service on the test server. Make sure to also send any Connect resources that have changed.
3. The test environment should now be ready for testing ***project X.***
4. **Note:** this also means that the test environment will be running only that project.
5. Once testing is complete, integrate the changed project with the other projects. This takes place in ***the development environment!***
6. First integrate the changed project with the others, and then deploy everything on the test server.

Integrating projects

To have multiple projects run on the same server, we will have to combine the individual project's Workflow configs into a single integrated Workflow config. This is done in the Integration project on the development environment.

It is important to do this in the development environment, because it involves manual changes, and these should travel through the entire DTAP process.

When all projects have been set up as above, creating the integrated Workflow config should be the only thing that is necessary.

Creating the integrated projects Workflow configuration

Let's assume there is a Workflow config with some processes and/or variables that are only needed when running multiple projects in a single Connect Workflow service. This would exist in the "`\OL Connect Solutions\Integration\Configurations\Workflow\`", and perhaps be named `base-integrated.OL-workflow`.

To create the integrated Workflow configuration, the following needs to happen:

- Open `base-integrated.OL-workflow` in Workflow editor
- Import every project's Workflow config as follows:
    - Use Import/Workflow/
    - Press "Select All'
    - Enable "Overwrite existing components with same name"

- Unselect Startup process
- Press OK
- Save as `integrated-solutions.OL-Workflow`

The `integrated-solutions.OL-Workflow` can now be sent to the local Workflow service on the development server for verification. Once it's done, everything can be promoted to the test server.
If all solutions obey to the rules laid out above, creating the integrated Workflow configuration can be done fast and without risk.

Promoting changes through DTAP

When projects are set up as described above, promoting changes from Development to, Test, Acceptance, and Production becomes easy:

- copy the "`\OL Connect Solutions\`" folders to the next environment;
- send any changed Connect resources to OL Connect Workflow;
- send the Workflow config of choice to the OL Connect Workflow

Auto deploying projects, and externalizing environment specific parameters is essential to make this work smoothly.

Instead of promoting changes by copying files, it's also an option to use a revision control system to manage your DTAP environment.

Using a Revision Control System - Git

For our sample solution development, we also use a revision control system to facilitate working with multiple people. Besides the obvious added control to manage changes, we also use this to further streamline our DTAP environment (DTP in our case, because we don't use Acceptance). We use Git for this, but the same can be done with Subversion, or any other modern revision control system.

Our procedure for working with Git :

- We use 3 permanent branches; master (the default for Git), test, and prod.
- Any changes are committed on the master branch from our development environments. If there are conflicting changes, they are resolved here.
- Once things are ready for testing, the master branch is merged to the test branch.
- To deploy changes in Test, we simply pull the test branch on the Test server, and then do whatever is needed to deploy the new version (send to Workflow, etc.)

- Once testing is complete, the test branch is merged to the prod branch.
- To deploy changes in Production, we pull the prod branch, and repeat what we did in the Test environment.

It's a is pretty simple process that suits our needs as a small team.

While testing, we often troubleshoot and debug directly on the Test server, but the easiest way to apply fixes, is to fix them in development, commit those on the master branch, and then merge again to test. If changes are committed to the test branch, merging them from test to master could be trickier than vice versa (with Git, that is).

If anything is wrong in Production, it's possible to pull the prod branch to a different environment so we can troubleshoot without disrupting Production. Small emergency fixes or work-arounds can be tested in Test, and directly applied in Production if needed, while a longer term solution can be prepared in Development.