Package management

Summary

Large binary files should not be included in source control. Consider using a package manager to archive, version, and install external dependencies. A package manager would also help support automated deployments.

Problem description

It is common for software projects to depend on other software developed outside the project. The current method for handling these files in the PSI project is to store them in the source control repository.

Large binary files adversely impact the performance of a git repository. Git cannot track only the changes to these files like it does for text. Each version of a binary file is effectively its own independent copy.

A recommended best practice is to use a package management solution for including external dependencies.

> As your team works with editors and tools to create and update files, you should put these files into Git so your team can enjoy the benefits of Git's workflow. Don't commit other types of files, such as DLLs, library files, and other dependencies that aren't created by your team but your code depends on into your repo. Deliver these files through package management to your systems.
>
> *What kind of files should you store in Git?*

Package managers

Package managers (also called package management systems) help store, version, archive, and install software. Many languages and development ecosystems have package management solutions.

- python has pip
- javascript has npm
- dotnet has nuget
- java has maven and gradle

When software is built, it can be packaged and stored in a package management repository with an explicit version number. Other developers can use the software by downloading and referencing it in their local projects. The local source code includes a file which specifies the project name and version. When the project requires a different version of a package, this file is updated and commited in the version control system. Build tools can automatically download packages when they are missing.

Goal: automate build, test, and deploy

For consistency, maintainability, and reliability we should strive to automate building, test, and deploying all software including the PSI project. To automate deployment, we must know the correct versions of all files required in an enviroment. If a binary dependency needs to be updated, that must be included in source control. Automated tools can use that information to identify the correct version of a file to use in a deployment package.

Using a package management system supports our goal of automating PSI deployments.

Azure Artifacts

Azure DevOps includes a organization controlled repository for storing packages called Azure Artifacts. Most major package managers are supported. It is an existing and available component in the ADO ecosystem that would be easily available from Azure Pipelines.

Proposed actions

Explore package management

A package management solution would require a change to our processes. Before making the change we should explore how to use the tools. They would require dependencies to be packaged and versioned. We must take the time to understand the impact and propose processes for packaging and installation.

Remove large binary files

Removing files in the next commit is not sufficient. Every copy of every large file is still retained in the history of the repository. Removing the files from history is not trivial. We will need to trial, test, and validate a process in a test repository before implementing in the live repository.

Define a dependency control solution

Whether we decide to use a package management solution or if we decide to store files on the file system, the process needs to be well defined.