Environment Configuration Updater Documentation

This utility was built to create a new Release using the VSTS Release Management API and populate specific environment and release variables.

Running the Utility:

Command line arguments:

Required Parameters

* releaseid
  + [int] Id of the release definition you want to create
* config
  + [string] File path to the environment configuration file, used to update environment variables
* buildversion
  + [string] Sets the Release level variable defining the build version of eRest being built

Optional Parameters

* istest
  + [bool] determines if the api is actually hit, this will create all the calls and log them to a file. Defaults to false
* vstscollection
  + [string] sets the collection portion of the url used to hit the api. Defaults to ‘DefaultCollection’
* vstsdomain
  + [string] sets the domain portion of the url used to hit the api. Defaults to ‘newmarketrm.vsrm.visualstudio.com’
* vststeamproject
  + [string] sets the team project portion of the url used to hit the api. Defaults to ‘NWS\_eRest’
* apiversion
  + [string] sets the api version of the release management api. Defaults to the api version defined in the app.config. (currently -2.2-preview.1)
* output
  + [string] sets the file path for the log that gets generated based on the apps output. Used primarily for debugging purposes. Defaults to the current directory with the name [Day][Month][Year]\_[Hour][Minute][Second]\_log.txt

Example of command

**EnvironmentConfigUpdater.exe releaseid 1 config EnvironmentConfig.json buildversion 1.2.3.4**

**EnvironmentConfigUpdater.exe [paramid] [paramvalue] ….**

**Flow of API Calls**

1. GET ReleaseDefinition
   1. Use the ReleaseDefinitionId provided via the command line
   2. Use the response to determine the highest rank environment and use that value when creating a new ReleaseDraft
2. POST DraftRelease
   1. Specify the highest ranked environment and the Release Definition Id
   2. This will return a Release with a list of environments, here is where we will update all of the environment variables to be whatever is provided in the EnvironmentConfig. This is a full overwrite operation
   3. Set the release level variables, in this case its only going to be the provided version of eRest
   4. When finding the environment variables in the config if the environment name contains a ‘-‘ it will look for an environment matching the value before the ‘-‘. If it doesn’t contain that character it will match the name identically
      1. NWSDev-123Server would search the config file for an environment with name NWSDev
      2. ThisSpecialEnvironment would search the config file for an environment with name ThisSpecialEnvironment
      3. Environment names are case-sensitive.
3. POST Release
   1. This saves the release with all the updated environment variables
4. GET Release
   1. Gets the release you have just updated, with all the latest environment variables
   2. This will also get you the latest list of artifact sources
5. GET ArtifactVersions
   1. Use the artifact sources from the previous get to retrieve all the available versions for each artifact.
   2. Get the highest version of each artifact and save that value.
6. POST Release
   1. Before posting this release you will need to update the following portions of the payload
      1. Set the Version and Branch location of all the Artifacts Sources in the release
      2. Set the release status to 1 and the release reason to 1
      3. For every environment set the environment status to 0 and null out both the Original Deploy Approvals
7. PATCH Release
   1. Send a patch to the same endpoint used above, the payload will be nothing more than ‘status = 2’. This will actually start the release.