EdkRepo User Guide

Jun 28, 2019

CONTENTS

EdkRepo is built on top of git. It is intended to automate common developer workflows for projects that use more than one git repository. For example many of the new projects in the edk2-platforms repository require the user to clone several git repositories. EdkRepo makes it easier to set up and upstream changes for these projects. EdkRepo does not replace git, rather it provides higher level extensions that make it easier to work with git. EdkRepo uses "manifest XML" files to describe the git repositories and EDK II packages that a project uses. EdkRepo is aware of the DSC file format and is able to combine this XML with metadata from DSC files to create a development experience tailored specifically for EDK II firmware development. EdkRepo is written in Python and is compatible with Python 3.5 or later.

CHAPTER

ONE

COMMAND REFERENCE

The commands provided by EdkRepo are documented below. These commands provide functionality for downloading and setting up a local workspace, syncing workspaces and moving between predefined groups of branches. Additional commands and command line arguments may be added over time to add functionality and improve user experience.

EdkRepo is meant to be used in conjunction with git. Tasks not automated with EdkRepo are meant to be performed with git.

1.1 Common Options

The following options are available for all commands descirbed in this document. Thus thet are not repeated in the details for each command.

- *-v / - -verbose*: Enables verbose output mode.
- *-h / - -help*: Display the help message for a specific command.
- *-o / - -override*: Ignore warning messages and proceed with the selected action.

1.2 Checkout

The checkout command enables checking out a predefined groups of branches also known as a branch combination. These branch combinations are defined in the project manifest file and represent combinations of repositories and their associated branches that are of relevance for the project. The checkout command also enables the user to recreate the state of their workspace at a specified moment in time by looking at the SHA of a specified commit. In this case EdkRepo updates the repository containing the provided SHA to this commit and attempts to calculate and update all remaining repositories in the workspace with commits that were current at the time of the provided SHA. In this case all repositories will be left in detached head mode.

Synopsis

```
edkrepo checkout <Combination or SHA>
```

Options

- *Combination or SHA*: The name of the combination as defined in the workspace manifest file to checkout or the SHA of the revision to checkout.

1.3 Clone

The clone command creates a local workspace on the user's system. Using the project manifest file the clone commands pulls all listed repositories and checks out the selected branch combination. Cloning a project also appropriately initializes submodules which may be contained within the listed repositories. Additionally the clone command has the ability to install client side Git hooks if listed in the project manifest.

A sparse checkout option is available to limit the user's workspace to only the source code required for a given project. The sparse checkout settings are defined in the project manifest file.

Clone can accept either a project name or the path to a manifest file. If a project name is specified by the user the clone command will search for a project with that name in the CiIndex file located in the global manifest repository. If the specified project is found clone will use the project manifest file indicated in the CiIndex file. Note: the location of the global manifest file on the user's system is dependent on both the operating system and the contents of the edkrepo.cfg file. Please see the *Application Data Directory* and *Configuration File* sections for more information.

If a project name is provided but is not found in the CiIndex file then clone will search for a manifest file with the specified name using the following search pattern:

1. Project Name in CiIndex.xml
2. If an absolute path is provided, the manifest file at that path.
3. If a relative path is provided then:

- (a) Search relative to the current working directory.
- (b) Search relative to the global manifest repository.

Synopsis

```
edkrepo clone <Workspace> <ProjectNameOrManifestFile> [Combination] [--sparse]
[--nosparse] [--skip-submodule]
```

Options

- *Workspace*: An empty directory to clone into; "." Indicates the current working directory
- *ProjectNameOrManifestFile*: Either the name of the project to pull or a path to a manifest file. See the clone process description above for details on how a manifest file is located. New projects will be added to the manifest repository and the Ciindex.xml file by the project/manifest owner.
- *Combination*: The branch combination to check out. If not specified the default combination as defined in the project manifest file will be used.
- - *-sparse*: Indicates that a sparse checkout is being requested. This will run a dependency check for the platform and determine what packages and modules need to be present to build. Only the required modules are then checked out, producing a sparse but clean tree. If the sparseCheckout setting in the project manifest file is set to false this flag has no effect.
- - *-nosparse*: Indicates that the sparse checkout feature is to be disabled. This flag will have no effect if the sparseCheckout setting in the project manifest file is set to False.
- - *-skip-submodule*: Indicates that submodule initialization, sync and update operations should be skipped.

1.4 Combo

Displays all branch combinations that are defined in the project manifest file. The currently checked out combination will be highlighted in green and displayed with a * in front of the name.

Using the *-v/- -verbose* flag with the combo command will list the repository and branch information for each branch combination defined in the project manifest file.

Synopsis

```
edkrepo combo [-c, - -color]
```

Options

- *-c / - - color*: Force color output

1.5 Manifest

The manifest command lists, by project name, all available projects located in the global manifest repository. Basic verification of these project manifest files is also conducted. An archived flag allows the user to see any projects which may have been marked as archived.

Synopsis

```
edkrepo manifest [-a, - -archived] [-c, - -color]
```

Options

- *-a, - -archived*: Include archived projects.
- *-c, - -color*: Force color output.

1.6 Sparse

The sparse command allows the user to determine and change the current project's sparse checkout status. The available sparse checkout settings for a project are determined by the project manifest file.

Synopsis

'edkrepo sparse [- -enable] [- -disable]

Options

- - *-enable*: Enables sparse checkout on a project with sparse checkout support.
- - *-disable*: Disables sparse checkout on a project.

1.7 Sync

The sync command updates the user's workspace including source code and any edkrepo managed configurations. The sync operation is based off of the local copy of the project manifest file stored in the /repo directory. Additionally the tool will update the global manifest repository and inform the user if there are any changes to the project manifest file available for download.

The local copy of each branch as listed in the currently checked-out branch combination is updated with the latest upstream changes; any submodules that are part of the target repositories are synced as appropriate as well. Any local branches present on the user's system are not modified by this command. Desired rebasing of local branches is expected to be completed by the user following the completion of the sync command using git rebase. If a dynamic sparse checkout has been performed, the dependency checker will be run to see if any updates to the sparse tree need to be made.

The fetch option retrieves the latest changes but does not update the local copies of the target branches.

User's may update their local copy of the project manifest file using the –update-local-manifest flag. The sync operation will be aborted and the user notified if any of the following conditions occur during the project manifest update:

- If there have been any changes to a repository's URL.
- If the current checked out combination is not present in the updated manifest.

Synopsis

```
edkrepo sync [– –fetch] [–u, – –update–local–manifest] [– –skip–submodule]
```

Options

- - *-fetch*: Performs only git fetch operations to get the latest content allowing the user to choose when and how to merger and / or rebase the content to their working directory.
- *-u* / - *-update-local-manifest*: Updates the local project manifest file found in the /repo directory prior to performing sync operations. All sync operations are then based on the updated project manifest file.
- - *-skip-submodule*: Indicates that submodule maitenance operations should be skipped.

CHAPTER

TWO

EDKREPO APPLICATION DATA DIRECTORY

In order to store various settings, configuration, and simply to have a global scratchpad directory, edkrepo will have a global data directory which is accessible to edkrepo from any user logged in to the current machine.

The location of this directory is determined by the OS and is described in the following sections.

2.1 Windows

The EdkRepo Application Data Directory can be found in C:\ProgramData\edkrepo.

This path can be found with the following Win32 API call:

- `SHGetFolderpath()` with `CSIDL_COMMON_APP_DATA`
- Use `ctypes` to access this API

2.2 macOS

The EdkRepo Application Data Directory can be found in /Library/Application Support/edkrepo.

This path can be found with the following API call:

- `NSSearchPathForDirectoriesInDomains()` using `NSApplicationSupportDirectory` for the `searchPath` and `NSLocalDomainMask` for the domain.
- Use `PyObjC` to access this API

Note: While the location of the EdkRepo Application Data Directory for macOS is described in this document full support for macOS is still a work in progress. At the current time EdkRepo will thow an OSError excection to indicate that support is currently incomplete.

2.3 Linux

The EdkRepo Application Data Directory can be found in /var/lib/edkrepo in conformance with the Filesystem Hierarchy Standard

CHAPTER

THREE

EDKREPO GLOBAL CONFIGURATION FILE

The EdkRepo global configuration file is stored in the *EdkRepo Application Data Directory*. It is an INI formatted file describing required configuration data. The placement of this configuration file is controlled by the EdkRepo installer. All contents are required for Edkrepo functionality, an exception will be raised if content is missing.

An example of this file and definitions for each section are included below:

```
[manifest-repo]
URL=
Branch=
LocalPath=

[git-ver]
minimum = 2.13.0
recommended =

[sparsecheckout]
always_include=
always_exclude=

[f2f-cherry-pick]
ignored_folder_substrings =
```

3.1 [manifest-repo]

This section provides information about the location of the global manifest repository used by Edkrepo.

URL

This is the URL of the global manifest repository.

Branch

The specific branch of the global manifest repository to pull.

LocalPath

The location that the manifest repository should be placed on the local machine. If a relative path is provided then it should be relative to the *EdkRepo Application Data Directory*.

3.2 [git-ver]

This section provides information regarding the minimum and suggested version of git which should be installed for optimal operation of EdkRepo.

minimum

This is the minimum version of git required for EdkRepo functionality. EdkRepo uses features of git that were not released until this version.

recommended

This is the version of git that EdkRepo has been validated with and is known to be working.

3.3 [sparsecheckout]

This section is deprecated and only maintained for backwards compatibility. Sparse checkout settings should now be described in the manifest file. The '|' character is used as a directory name separator.

always_include

A list of directories to always be included when sparse checkout is enabled.

always_exclude

A list of directories to always exclude when sparse checkout is enabled.

3.4 [f2f-cherry-pick]

This section is reserved for future functionality and should be left blank.

ignored_folder_substrings

This section is reserved for future functionality and should be left blank.

CHAPTER

FOUR

EXIT CODES

EdkRepo will return the following exit codes back to the operating system shell depending on the result of the requested operation.

- *0*: The command completed successfully
- *1*: A general error occurred which does not meet any of the other error descriptions in this document
- *101*: Invalid Parameter: One or more of the command line parameters provided are not valid for the requested operation
- *102*: Global Configuration Not Found: The global configuration file was not found in the application data directory. Please see section 4 for more details.
- *103*: Configuration File Invalid: The global configuration file has invalid data or syntax errors.
- *104*: Manifest Invalid: One or more manifest files has invalid data or syntax errors.
- *105*: Workspace Invalid: The directory is not a valid edkrepo workspace.
- *106*: Global Data Directory Not Found: The edkrepo global data directory was not found. Please see section 4 for more details.
- *107*: Uncommitted Changes: There are uncommitted or unstaged changes present in the repository that must be handled before proceeding with the requested operation.
- *108*: Manifest Not Found: A manifest for the selected project was unable to be found in the global manifest repository.
- *109*: Manifest Changed: The contents of the manifest file have changed in such a way that the requested operation could not be completed.
- *110*: WIT Not Found: The WIT executable was not found
- *111*: Reviewers: No reviewers were found so the requested review could not be sent.
- *112*: Config File Read Only: The requested operation was unable to be completed because the config file was read only.
- *114*: Verification Failed: The verification of the manifest repository and associated manifest files has failed.
- *115*: Sparse Checkout Failed: An invalid combination of actions have been requested for the current project.
- *116*: Not Found
- *117*: Found Multiple
- *123*: Warning
- *129*: A Git exception has occurred.
- *130*: The Git hooks specified in the manifest file were not found.

- *131*: There was an error setting up the git configuration information.

CHAPTER

FIVE

CIINDEX.XML

The CiIndex.xml file is used by EdkRepo to determine whether a project exists in the global manifest repository, to resolve that projects location relative to the root of the global manifest repository, and to present a list of all active and archived projects available to users of EdkRepo via the *Manifest command*.

The CiIndex.xml file consists of a *ProjectList* which contains one or more *Project* entries as shown in the below example:

```
<ProjectList>
  <!-- The 'xmlPath' values should be relative to this file (CiIndex.xml) -->
  <Project name="SampleProject" xmlPath="SampleProj\SampleManifest.xml" />
  <Project name="SampleProject2" xmlPath="SampleProj2\SampleManifest2.xml" archived=
↪"true" />
</ProjectList>
```

The name listed for a given project **MUST** match the code name in the project manifest file. otherwise EdkRepo will flag it as an invalid manifest. The *archived* field is optional.

CHAPTER

SIX

PROJECT MANIFEST FILE

The project manifest file, stored in the workspace as repo/Manifest.xml or in the Global Manifest Repository under the path described for the project in the *CiIndex.xml* file, describes the source contents, build configurations, sparse checkout settings, and other important information about the project. The project manifest file is used by Edkrepo to perform most operations. All described sections are considered required unless stated otherwise. If all required sections are not present the project manifest file will be considered invalid.

The example manifest below is the IntelMinPlatformManifest.xml:

```
<Manifest>
  <ProjectInfo>
    <CodeName>Intel-MinPlatform</CodeName>
    <Description>Intel Minimum Platform</Description>
    <DevLead>michael.a.kubacki@intel.com</DevLead>
    <LeadReviewers>
      <Reviewer>michael.a.kubacki@intel.com</Reviewer>
      <Reviewer>chasel.chiu@intel.com</Reviewer>
      <Reviewer>nathaniel.l.desimone@intel.com</Reviewer>
      <Reviewer>ankit.sinha@intel.com</Reviewer>
      <Reviewer>liming.gao@intel.com</Reviewer>
      <Reviewer>ray.ni@intel.com</Reviewer>
      <Reviewer>rangasai.v.chaganty@intel.com</Reviewer>
      <Reviewer>isaac.w.oram@intel.com</Reviewer>
      <Reviewer>daocheng.bu@intel.com</Reviewer>
      <Reviewer>shifei.a.lu@intel.com</Reviewer>
      <Reviewer>bowen.zhou@intel.com</Reviewer>
    </LeadReviewers>
  </ProjectInfo>

  <GeneralConfig>
    <PinPath>edk2-platforms/Intel-MinPlatform/pins</PinPath>
    <DefaultCombo combination="master" />
    <CurrentClonedCombo combination="master" />
  </GeneralConfig>

  <SparseCheckout>
    <SparseSettings sparseByDefault="true" />
    <SparseData>
      <!-- Always include all files in the repository root. -->
      <AlwaysInclude>*.*</AlwaysInclude>
    </SparseData>
    <SparseData remote="Edk2Repo">
      <AlwaysInclude>BaseTools|Conf</AlwaysInclude>
      <AlwaysInclude>FatPkg|IntelFsp2Pkg|IntelFsp2WrapperPkg</AlwaysInclude>
      <AlwaysInclude>MdeModulePkg|MdePkg|NetworkPkg|PcAtChipsetPkg|SecurityPkg</
↪AlwaysInclude>
```

(continues on next page)

(continued from previous page)

```
      <AlwaysInclude>ShellPkg|SourceLevelDebugPkg|UefiCpuPkg|FmpDevicePkg|CryptoPkg</
↪AlwaysInclude>
    </SparseData>
    <SparseData remote="Edk2PlatformsRepo">
      <AlwaysInclude>Platform/Intel/AdvancedFeaturePkg|Platform/Intel/
↪MinPlatformPkg|Platform/Intel/BoardModulePkg</AlwaysInclude>
      <AlwaysInclude>Platform/Intel/DebugFeaturePkg|Platform/Intel/
↪UserInterfaceFeaturePkg|Platform/Intel/ClevoOpenBoardPkg</AlwaysInclude>
      <AlwaysInclude>Platform/Intel/KabylakeOpenBoardPkg|Platform/Intel/
↪PurleyOpenBoardPkg|Platform/Intel/Readme.md</AlwaysInclude>
      <AlwaysInclude>Platform/Intel/build.cfg|Platform/Intel/build_bios.py</
↪AlwaysInclude>
      <AlwaysInclude>Silicon/Intel/IntelSiliconPkg|Silicon/Intel/
↪KabylakeSiliconPkg|Silicon/Intel/LewisburgPkg</AlwaysInclude>
      <AlwaysInclude>Silicon/Intel/PurleyRcPkg|Silicon/Intel/PurleySktPkg</
↪AlwaysInclude>
    </SparseData>
    <SparseData remote="Edk2NonOsiRepo">
      <AlwaysInclude>Silicon/Intel/KabylakeSiliconBinPkg|Silicon/Intel/
↪PurleySiliconBinPkg</AlwaysInclude>
    </SparseData>
    <SparseData remote="FspRepo">
      <AlwaysInclude>AmberLakeFspBinPkg|CoffeeLakeFspBinPkg|KabylakeFspBinPkg</
↪AlwaysInclude>
    </SparseData>
  </SparseCheckout>

  <RemoteList>
    <!-- The 'name' attribute for each Remote tag must be unique. -->
    <Remote name="Edk2Repo">https://github.com/tianocore/edk2.git</Remote>
    <Remote name="Edk2PlatformsRepo">https://github.com/tianocore/edk2-platforms.git</
↪Remote>
    <Remote name="Edk2NonOsiRepo">https://github.com/tianocore/edk2-non-osi.git</
↪Remote>
    <Remote name="FspRepo">https://github.com/IntelFsp/FSP.git</Remote>
  </RemoteList>

  <ClientGitHookList>
  </ClientGitHookList>

  <CommitTemplates>
  </CommitTemplates>

  <SubmoduleAlternateRemotes>
  </SubmoduleAlternateRemotes>

  <CombinationList>
    <Combination name="master" description="the master branch from edk2 Repo, master␣
↪branch from edk2-platforms repo, devel-MinPlatform branch from edk2-non-osi repo,␣
↪master branch from FSP repo">
      <Source localRoot="edk2" remote="Edk2Repo" branch="master" sparseCheckout="true
↪" enableSubmodule="true" />
      <Source localRoot="edk2-platforms" remote="Edk2PlatformsRepo" branch="master"␣
↪sparseCheckout="true" />
      <Source localRoot="edk2-non-osi" remote="Edk2NonOsiRepo" branch="devel-
↪MinPlatform" sparseCheckout="true" />
      <Source localRoot="FSP" remote="FspRepo" branch="master" sparseCheckout="true" /
↪>
```

(continues on next page)

(continued from previous page)

```
    </Combination>
  </CombinationList>

  <DscList>
    <!-- List of the top-level Dsc's that are used to build this Platform.
         These files will be parsed by EdkRepo to determine pkg dependencies
     -->
  </DscList>
</Manifest>
```

6.1 ProjectInfo

This section contains information used to identify the project its key developers and reviewers.

CodeName

The name of the project described by this project manifest file. The content of this field must exactly match the project name associated with this project manifest in the *CiIndex.xml* file.

Description

A short description of the project defined by this manifest file.

DevLead

A list of zero or more emails for the lead developers or maintainers of the project described by the manfiest file. The presence of this field is not required.

LeadReviewers

A list of zero or more `<Reviwer></Reviewer>` items. The presence of this field is not required.

6.2 GeneralConfig

This section contains basic configuration settings of the project used by EdkRepo.

DefaultCombo

The DefaultCombo indicates which combination will be checked out by default if the *clone command* is run without the Combination parameter to override it.

CurrentClonedCombo

The current cloned combo field is intended to be used by EdkRepo to store workspace state; the copy of the project manifest in the global manifest repository should have this value set to the same value as the default combo while EdkRepo will manipulate the copy of the project manifest stored in the users workspace so that this value is that of the combination which is currently checked out.

6.3 SparseCheckout

The sparse checkout section specifies data used by the *sparse command* and other commands to determine which files and folders are used by a project. Sparse checkout allows the developer to limit their view of the repositories contained in their workspace to only the files and folders that are applicable to the project that the developer is working on.

This presence of this section is not required if sparse checkout functionality is not desirable for the described project.

SparseSettings

Project dependencies must be explicitly specified in the SparseData sections using the <AlwaysInclude></AlwaysInclude> and <AlwaysExclude></AlwaysExclude> tags. Setting this attribute to false has a significant performance benefit.

6.4 RemoteList

The remote list section describes the repositories which are available for inclusion in the projects branch combinations.

Remote

The name= attribute indicates the name that the remote will be referred to by other sections of the project manifest file, this attribute must be unique. The remote also contains the git URL of the remote repository.

6.5 ClientGitHookList

The client git hook list section contains the source and destination information for any client side hooks needed by the projects remote repositories. The source can be either a URL or a global manifest repository relative path. These client side git hooks are downloaded and installed by EdkRepo whenever a repository is cloned.

If no hooks are needed by the project this section may be omitted from the project manifest file.

ClientGitHook

The source= attribute indicates the URL to retrieve the hook from. The remote= attribute indicates the repository that the hook is for and must exactly match the name= attribute of a remote listed in the <RemoteList> section of the project manifest file. The destination= attribute is the **workspace** relative location that EdkRepo will copy the hook to when cloning the repository.

6.6 CombinationList

The Combination List section of the project manifest file lists the branch combinations which have been defined for the project. At least one combination must be defined in this section. Currently asymmetric combinations are not supported by EdkRepo so all combinations must reference the same set of remote repositories.

Combination

The <Combination> subsection of the Combination List describes and individual branch combination for the project. The name= attribute indicates the name of the combination and is used by EdkRepo for operations such as clone and checkout and must be unique for each combination described in the Project Manifest file. The description= attribute is meant to briefly describe the purpose of the combination. Additionally each combination consists of at least one <Source> subsection

Source

The Source subsection describes the sources used to create the branch combination. The localRoot= attribute indicates the WORKSPACE relative location that the source code will be downloaded to by EdkRepo during a clone operation. The remote= attribute is a reference to the remote repository from which code should be pulled and must exactly match the name attribute of a Remote in the RemoteList section of the Project Manifest file. The branch= attribute indicates which branch to pull from the selected remote repository. The sparseCheckout= attribute indicates whether sparse checkout is enabled by default for the project and can be overridden by the user during edkrepo

clone command or via the sparse command. The `enableSubmodule=` attribute indicates whether pulling/syncing submodules is enabled for the project. Both `sparseCheckout` and `enableSubmodule` are not required and default to false.

6.7 SubmoduleAlternateRemotes

The Submodule Alternate Remotes section of the project manifest file is optional. This section lists the alternate remote URLs to enable URL redirection for git repositories. Each entry in this section needs to list the original URL, a remote name that matches one from the Remote List section and an update URL that it will be redirected to.

EdkRepo will use the information in this section to create a conditionally included git configuration file stored in the workspace/repo directory. This file will be named .gitconfig-repo-name. This file will include an InsteadOf entry directing git to use the alternate URL instead of the original URL. Edkrepo will also create a conditional include directive in the users global git configuration file indicating that the generated configuration is used for only the repository which is defined for.

Note: a minimum Git version of 2.13 is required to use the feature.

6.8 DscList

The DscList section contains a list of the DSC files used to build the platform described by the Project Manifest file. If the useDsc attribute in the SparseCheckout section of the Project Manifest file is true the DSC files listed in this section are used to compute the sparse checkout and project filter settings.

Dsc

The Dsc subsections contains a **workspace** relative path to the specified DSC file.