Package 'vstsr'

March 14, 2018

Type Package

Title Access to Visual Studio Team Services API via R

Version 1.0.0

Description Implementation of Visual Studio Team Services <https://www.visualstudio.com/team-services/> API calls.
It enables the extraction of information about repositories, build and release definitions and individual releases.
It also helps create repositories and work items within a project without logging into Visual Studio.
There is the ability to use any API service with a shell for any non-predefined call.

License GPL-2

URL https://github.com/ashbaldry/vstsr,
https://docs.microsoft.com/en-us/rest/api/vsts

BugReports https://github.com/ashbaldry/vstsr/issues

Imports R6, httr, RCurl, magrittr, jsonlite, xml2

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests testthat

NeedsCompilation no

Author Ashley Baldry [aut, cre]

Maintainer Ashley Baldry <arbaldry91@gmail.com>

Repository CRAN

Date/Publication 2018-03-14 10:20:42 UTC

R topics documented:

`vstsr` *vstsr: A package connecting R and Visual Studio*

Description

This package takes a look at the Visual Studio Team Services API calls and wraps them around in easy to use R functions. This includes looking at projects, repositories, work items, and eventually sections such as builds and releases.

Details

For more information about Visual Studio Team Services APIs, take a look at `https://docs.microsoft.com/en-us/rest/api/vsts`

Projects

This looks at the projects available in a visual studio instance. This lets you change between any project that you might have access to.

Repositories

This looks primarily at the Git repositories available, and whether you want to create a new repository or delete an existing one.

Work Items

This will track any existing work item for a project, and also the ability to create a new work item for a project. Useful if working within a development team and automate inclusion of creating bugs upon creating any certain R error.

Releases

This will look at the releases available for a project, both the definitions and the actual releases. There is the ability to deploy the created releases to a new environment.

Custom

For any non-predefined API service available in the package, it is always possible to run the `vsts_run_command` which will use any enabled Visual Studio API call.

`vsts_account` *Visual Studio Team Services Account*

Description

Visual Studio Team Services Account

Usage

```
vsts_account
```

Format

An `R6Class` generator object

Details

For the majority of functions that are within this `vsts_account` object, you can get help about the query or body parameter with `?vsts_<function name>`.

Fields

`user` username for the Visual Studio account

`pass` password for the Visual Studio account

`domain` domain name for the Visual Studio account

`project` (optional) project name within the domain of the Visual Studio account

`repo` (optional) repository name with the project of the Visual Studio domain

Examples

```
## Not run:
proj <- vsts_account$new('<username>', '<password>', '<domain>',
                         '<project>', '<repo>')
str(proj)

## End(Not run)
```

`vsts_auth_key` *Visual Studio Team Service Authentication Key*

Description

Creation of a VSTS authentication key that will be used when running any of the API calls.

Usage

```
vsts_auth_key(user, pass)
```

Arguments

`user`	username to access VSTS project
`pass`	password to access VSTS project

Value

An authentication key string in the form of 'Basic <Base 64 of user:pass>'

Examples

```
auth_key <- vsts_auth_key('<username>', '<password>')
```

`vsts_create_release` *Visual Studio Project Release Information*

Description

These functions will allow you to create releases from Visual Studio.

Usage

```
vsts_create_release(domain, project, auth_key, body)
```

Arguments

`domain`	the location of the visual studio server
`project`	the name of the project in `domain` to look at
`auth_key`	authentication key generated by using `vsts_auth_key`
`body`	a list of extra parameters that can need to be sent to the API call (* mandatory): `artifacts *` [list] Sets list of artifact to create a release. Check `Details` for more information.

`definitionId` * [integer] Sets definition Id to create a release.

`description` * [character] Sets description to create a release.

`isDraft` [logical] Sets 'true' to create release in draft mode, 'false' otherwise.

`manualEnvironments` [character] Sets list of environments to manual as condition.

`properties` [list] The class represents a property bag as a collection of key-value pairs.

`reason` [character] Sets reason to create a release.

Details

The `artifacts` object within the body contains two items:

- alias[character] Sets alias of artifact.
- instanceReference[list] Sets instance reference of artifact. e.g. for build artifact it is build number.

For more information about release API calls check `https://docs.microsoft.com/en-us/rest/api/vsts/release/releases`.

Examples

```
#Add in own details to get a non-NULL output
auth_key <- vsts_auth_key('<username>', '<password>')
art_list <- list(list(alias = 'Art1', instanceReference = list(id = 1)),
                 list(alias = 'Art2', instanceReference = list(id = 2)))
body <- list(definitionId = 1, description = 'R API Release',
             artifacts = I(art_list))
vsts_create_release('domain', 'project', auth_key, body)
```

`vsts_create_workitem` *Visual Studio Project Work Items*

Description

These functions will allow you to scrape work item information from a particular Visual Studio project.

Usage

```
vsts_create_workitem(domain, project, item_type, auth_key, ...)
```

Arguments

`domain`	the location of the visual studio server
`project`	the name of the project in `domain` to look at
`item_type`	the type of work item to be created
`auth_key`	authentication key generated by using `vsts_auth_key`
`...`	arguments passed to vsts_get_workitem_fields

Details

For more information about work item API calls check `https://docs.microsoft.com/en-us/rest/api/vsts/wit/work%20items`.

`vsts_deploy_release` *Visual Studio Project Release Environment Information*

Description

These functions will allow you to run release environment tasks from Visual Studio.

Usage

```
vsts_deploy_release(domain, project, release, env, auth_key)
```

Arguments

`domain`	the location of the visual studio server
`project`	the name of the project in `domain` to look at
`release`	the release ID of the release
`env`	the release environment ID to release on
`auth_key`	authentication key generated by using `vsts_auth_key`

Details

For more information about release environment API calls check `https://docs.microsoft.com/en-us/rest/api/vsts/release/releases/update%20release%20environment`.

Examples

```
#Add in own details to get a non-NULL output
auth_key <- vsts_auth_key('<username>', '<password>')
vsts_deploy_release('domain', 'project', auth_key, 1, 1)
```

`vsts_get_build_defs` *Visual Studio Project Build Definition Information*

Description

These functions will allow you to scrape build definition information from Visual Studio.

Usage

```
vsts_get_build_defs(domain, project, auth_key, query = NULL)
```

Arguments

`domain`	the location of the visual studio server
`project`	the name of the project in `domain` to look at
`auth_key`	authentication key generated by using `vsts_auth_key`
`query`	a list of extra parameters that can be sent to the API call: **revision** [integer] The revision number to retrieve. If this is not specified, the latest version will be returned. **minMetricsTime** [string] If specified, indicates the date from which metrics should be included. **propertyFilters** [string] A comma-delimited list of properties to include in the results. **includeLatestBuilds** [logical]

Details

For more information about the build definition API calls check `https://docs.microsoft.com/en-us/rest/api/vsts/build/definitions`.

Examples

```
#Add in own details to get a non-NULL output
auth_key <- vsts_auth_key('<username>', '<password>')
vsts_get_build_defs('domain', 'project', auth_key)
```

`vsts_get_commits` *Visual Studio Project Git Repositories*

Description

These functions will allow you to scrape git repository information from Visual Studio.

Usage

```
vsts_get_commits(domain, project, repo, auth_key, query = NULL)
```

Arguments

`domain`	the location of the visual studio server
`project`	the name of the project in `domain` to look at
`repo`	the name of the repository in `project` to look at
`auth_key`	authentication key generated by using `vsts_auth_key`
`query`	a list of extra parameters that can be sent to the API call:

- `branch` [character] the name of a branch in the repository (cannot combine with `commit`)
- `commit` [character] the id of a commit in the repository (cannot combine with `branch`)
- `itemPath` [character] path of an item in the repository
- `committer` [character] name of the person who committed the change
- `author` [character] name of the author
- `fromDate` [Date] start date to search from
- `toDate` [Date] end date to search from

Details

For more information about git repository API calls check `https://docs.microsoft.com/en-us/rest/api/vsts/git/`.

Examples

```
#Add in own details to get a non-NULL output
auth_key <- vsts_auth_key('<username>', '<password>')
vsts_get_commits('domain', 'project', 'repo', auth_key)
```

`vsts_get_projects` *Visual Studio Projects*

Description

These functions will allow you to scrape project information from Visual Studio.

Usage

```
vsts_get_projects(domain, auth_key, quiet = FALSE)
```

Arguments

`domain`	the location of the visual studio server
`auth_key`	authentication key generated by using `vsts_auth_key`
`quiet`	logical whether want general running information from printing. Any issue with the API call will still show up if set to `TRUE`

Details

For more information about project API calls check `https://docs.microsoft.com/en-us/rest/api/vsts/core/projects`.

`vsts_get_releases` *Visual Studio Project Release Information*

Description

These functions will allow you to scrape releases from Visual Studio.

Usage

```
vsts_get_releases(domain, project, auth_key, query = NULL)

vsts_get_release(domain, project, release, auth_key)
```

Arguments

`domain`	the location of the visual studio server
`project`	the name of the project in `domain` to look at
`auth_key`	authentication key generated by using `vsts_auth_key`
`query`	a list of extra parameters that can be sent to the API call: `propertyFilters` [character] A comma-delimited list of extended properties to retrieve.

`tagFilter` [character] A comma-delimited list of tags. Only releases with these tags will be returned.

`isDeleted` [logical] Gets the soft deleted releases, if true.

`sourceBranchFilter` [character] Releases with given sourceBranchFilter will be returned.

`artifactVersionId` [character] Releases with given artifactVersionId will be returned. E.g. in case of Build artifactType, it is buildId.

`sourceId` [character] Unique identifier of the artifact used. e.g. For build it would be projectGuid:BuildDefinitionId, for Jenkins it would be JenkinsConnectionId:JenkinsDefinitionId, for TfsOnPrem it would be TfsOnPremConnectionId:ProjectName:TfsOnPremDefinitionId. For third-party artifacts e.g. TeamCity, BitBucket you may refer 'uniqueSourceIdentifier' inside vss-extension.json `https://github.com/Microsoft/vsts-rm-extensions/blob/master/Extensions`.

`artifactTypeId` [character] Releases with given artifactTypeId will be returned. Values can be Build, Jenkins, GitHub, Nuget, Team Build (external), ExternalTFSBuild, Git, TFVC, ExternalTfsXamlBuild.

`$expand` [character] The property that should be expanded in the list of releases.

`continuationToken` [integer] Gets the releases after the continuation token provided.

`$top` [integer] Number of releases to get. Default is 50.

`queryOrder` [character] Gets the results in the defined order of created date for releases. Default is descending.

`maxCreatedTime` [Date] Releases that were created before this time.

`minCreatedTime` [Date] Releases that were created after this time.

`statusFilter` [character] Releases that have this status.

`createdBy` [character] Releases created by this user.

`searchText` [character] Releases with names starting with searchText.

`definitionEnvironmentId` [integer] Releases from this release environment Id.

`definitionId` [integer] Releases from this release definition Id.

`releaseIdFilter` [character] A comma-delimited list of releases Ids. Only releases with these Ids will be returned.

`release`	Release Definition ID

Details

For more information about release API calls check `https://docs.microsoft.com/en-us/rest/api/vsts/release/releases`.

Examples

```
#Add in own details to get a non-NULL output
auth_key <- vsts_auth_key('<username>', '<password>')
vsts_get_releases('domain', 'project', auth_key)
```

`vsts_get_release_defs` *Visual Studio Project Release Definition Information*

Description

These functions will allow you to scrape release definition information from Visual Studio.

Usage

```
vsts_get_release_defs(domain, project, auth_key, quiet = FALSE)
```

Arguments

`domain`	the location of the visual studio server
`project`	the name of the project in `domain` to look at
`auth_key`	authentication key generated by using `vsts_auth_key`
`quiet`	logical whether want general running information from printing. Any issue with the API call will still show up if set to `TRUE`

Details

For more information about release definition API calls check `https://docs.microsoft.com/en-us/rest/api/vsts/release/definitions`.

Examples

```
#Add in own details to get a non-NULL output
auth_key <- vsts_auth_key('<username>', '<password>')
vsts_get_release_defs('domain', 'project', auth_key)
```

`vsts_get_repos` *Visual Studio Project Repositories*

Description

These functions will allow you to scrape project information from Visual Studio.

Usage

```
vsts_get_repos(domain, project, auth_key, quiet = FALSE)

vsts_create_repo(domain, project, repo, auth_key, quiet = FALSE)

vsts_delete_repo(domain, project, repo, auth_key, quiet = FALSE)
```

Arguments

`domain`	the location of the visual studio server
`project`	the name of the project in `domain` to look at
`auth_key`	authentication key generated by using `vsts_auth_key`
`quiet`	logical whether want general running information from printing. Any issue with the API call will still show up if set to `TRUE`
`repo`	the name of the repository in `project` to look at

Details

For more information about repository API calls check `https://docs.microsoft.com/en-us/rest/api/vsts/git/`.

Examples

```
#Add in own details to get a non-NULL output
auth_key <- vsts_auth_key('<username>', '<password>')

#Get repo list
vsts_get_repos('domain', 'project', auth_key)

#Create new repo
vsts_create_repo('domain', 'project', 'repo', auth_key)

#Delete existing repo
vsts_delete_repo('domain', 'project', 'repo', auth_key)
```

`vsts_get_workitems` *Visual Studio Project Get Work Items*

Description

These functions will allow you to scrape work item information from a particular Visual Studio project.

Usage

```
vsts_get_workitems(domain, auth_key, query = NULL)

vsts_get_workitem(domain, auth_key, id)
```

Arguments

`domain`	the location of the visual studio server
`auth_key`	authentication key generated by using `vsts_auth_key`
`query`	a list of extra parameters that can be sent to the API call:
	`ids` [character] a comma-separated list of up to 200 IDs of the work items to get
	`fields` [character] a comma-separated list of up to 100 fields to get with each work item. If not specified, all fields with values are returned. Calculated fields such as Attached File Count must be specifically queried for using this parameter.
	`asOf` [Date] gets the work items as they existed at this time
	`ErrorPolicy` [character] determines if the call will throw an error when encountering a work item (default behavior) that doesn't exist (`throw`) or simply omit it (`omit`)
`id`	(for single work item request) ID of the work item to retrieve

Details

For more information about work item API calls check `https://docs.microsoft.com/en-us/rest/api/vsts/wit/work%20items`.

`vsts_get_workitem_fields` *Visual Studio Work Item Fields*

Description

This contains all the fields required of any work item in a visual studio project and helps add/rename the fields of the selected work item.

Usage

```
vsts_get_workitem_fields(System.Title, System.Description, System.TeamProject,
  System.AreaPath, System.IterationPath, Microsoft.VSTS.Common.Priority, ...)
```

Arguments

`System.Title`	[character] title of the Visual Studio work item
`System.Description`	[character] description of the Visual Studio work item
`System.TeamProject`	[character] name of the Visual Studio project
`System.AreaPath`	[character] path of the Visual Studio work item

`System.IterationPath`	[character] name of the Visual Studio iteration path
`Microsoft.VSTS.Common.Priority`	[integer] priority of the work item - 1 to 4
`...`	other fields that might have been missed out originally

Details

For more information about work item fields API calls check https://docs.microsoft.com/en-us/rest/api/vsts/wit/fields.

`vsts_run_command` *Visual Studio Custom API Calls*

Description

For any requirement not currently in place in the `vstsr` package, then this function will allow you to use the relevant API call without any extra requirements.

For the most part it is just a shell of `VERB` but will have the `auth_key` set up already.

Usage

```
vsts_run_command(url, verb, auth_key, body = NULL, query = NULL)
```

Arguments

`url`	the URI of the API call to run
`verb`	name of the verb to use
`auth_key`	authentication key generated by using `vsts_auth_key`
`body`	check `VERB` for more details. If the object is a named list, then it will be transformed into a JSON string so that can be added to the call. Use https://docs.microsoft.com/en-us/rest/api/vsts/ to find out any required parameter for the body.
`query`	a list of extra parameters that can be sent to the API call. If not required then leave as `NULL`

Examples

```
## Not run:
auth_key <- vsts_auth_key('<username>', '<password>')
#Get commits of a repository
URL <- paste0('https://{accountName}.visualstudio.com/{project}',
              '/_apis/git/repositories/{repositoryId}/commits/',
              '{commitId}?api-version=4.1-preview')
vsts_run_command(URL, 'GET', auth_key)

## End(Not run)
```

Index