CloudSDK Documentation v1.5.2196.335

This document contains comprehensive documentation for all public functions in the CloudSDK, including detailed parameter descriptions, usage examples, and best practices.

Table of Contents

CloudSDK Initialization

```
from eecloud.cloudsdk import CloudSDK, SDKBase
from eecloud.models import *

pxc = CloudSDK(cli_path=r"c:\path\to\cli.exe")
```

Environment

Environment Status

Description: Check the current environment status

Response Structure:

- `statuses` (Optional[list[Environment_EnvironmentStatus]]): List of Statuses (list)

  ▶ Properties of `Environment_EnvironmentStatus`
  - `Url` (Optional[str]): Url value
  - `Status` (Optional[int]): Status value
  - `Message` (Optional[str]): Message value
  - `Exception` (Optional[str]): Exception value

```
environment_environment_status_resp:
list[CommandResponse[Contracts_EnvironmentStatusResponse]] =
pxc.environment.environment_status(print_message=True)
environment_environment_status_final: Contracts_EnvironmentStatusResponse =
SDKBase.get_response_data(environment_environment_status_resp)

if environment_environment_status_final is not None:
    # Iterate over statuses list
    if environment_environment_status_final.statuses is not None:
        for item in environment_environment_status_final.statuses:
            # Access properties of Environment_EnvironmentStatus object
            print(f"Url: {item.Url}")
            print(f"Status: {item.Status}")
            print(f"Message: {item.Message}")
            print(f"Exception: {item.Exception}")
    else:
        print(f"No statuses returned")
else:
    print(f"environment_status failed:
{environment_environment_status_resp.Message}")
```

Get User Environment

Description: Get the current user environment configuration

Response Structure:

- `Environment` (Optional[str]): Environment value (single)

```
environment_get_user_environment_resp:
list[CommandResponse[Contracts_EnvironmentResponse]] =
pxc.environment.get_user_environment(print_message=True)
environment_get_user_environment_final: Contracts_EnvironmentResponse =
SDKBase.get_response_data(environment_get_user_environment_resp)

if environment_get_user_environment_final is not None:
    # Access single properties
    if environment_get_user_environment_final.Environment is not None:
        print(f"Environment:
{environment_get_user_environment_final.Environment}")
else:
    print(f"get_user_environment failed:
{environment_get_user_environment_resp.Message}")
```

Set User Environment

Description: Set the user environment for the current session

Parameters:

- `environment` (str): Environment name to connect to

Response Structure:

- `Environment` (Optional[str]): Environment value (single)

```
environment = "NA"

environment_set_user_environment_resp:
```

```
list[CommandResponse[Contracts_EnvironmentResponse]] =
pxc.environment.set_user_environment(environment=environment, print_message=True)
environment_set_user_environment_final: Contracts_EnvironmentResponse =
SDKBase.get_response_data(environment_set_user_environment_resp)

if environment_set_user_environment_final is not None:
    # Access single properties
    if environment_set_user_environment_final.Environment is not None:
        print(f"Environment:
{environment_set_user_environment_final.Environment}")
else:
    print(f"set_user_environment failed:
{environment_set_user_environment_resp.Message}")
```

Show Log File Path

Description: Display the folder path where log files are saved

Parameters:

- `todays_file` (bool): Show today's log file path (optional flag) (optional)

Response Structure:

- `LoggingPath` (Optional[str]): Loggingpath value (single)

```
todays_file = True

environment_show_log_file_path_resp:
list[CommandResponse[Contracts_ShowLogFileResponse]] =
pxc.environment.show_log_file_path(todays_file=todays_file, print_message=True)
environment_show_log_file_path_final: Contracts_ShowLogFileResponse =
SDKBase.get_response_data(environment_show_log_file_path_resp)

if environment_show_log_file_path_final is not None:
    # Access single properties
    if environment_show_log_file_path_final.LoggingPath is not None:
        print(f"LoggingPath: {environment_show_log_file_path_final.LoggingPath}")
else:
    print(f"show_log_file_path failed:
{environment_show_log_file_path_resp.Message}")
```

Auth

Check Authentication Status

Description: Check if user is currently authenticated

Response Structure:

- `IsAuthenticated` (Optional[bool]): Isauthenticated value (single)

- `Environment` (Optional[str]): Environment value (single)
- `UserName` (Optional[str]): Username value (single)
- `TenantName` (Optional[str]): Tenantname value (single)

```
auth_check_authentication_status_resp:
list[CommandResponse[Contracts_CheckAuthenticationStatusResponse]] =
pxc.auth.check_authentication_status(print_message=True)
auth_check_authentication_status_final:
Contracts_CheckAuthenticationStatusResponse =
SDKBase.get_response_data(auth_check_authentication_status_resp)

if auth_check_authentication_status_final is not None:
    # Access single properties
    if auth_check_authentication_status_final.IsAuthenticated is not None:
        print(f"IsAuthenticated:
{auth_check_authentication_status_final.IsAuthenticated}")
    if auth_check_authentication_status_final.Environment is not None:
        print(f"Environment:
{auth_check_authentication_status_final.Environment}")
    if auth_check_authentication_status_final.UserName is not None:
        print(f"UserName: {auth_check_authentication_status_final.UserName}")
    if auth_check_authentication_status_final.TenantName is not None:
        print(f"TenantName: {auth_check_authentication_status_final.TenantName}")
else:
    print(f"check_authentication_status failed:
{auth_check_authentication_status_resp.Message}")
```

Login Client Credentials

Description: Authenticate using client credentials

Parameters:

- `use_client_credentials` (bool): use_client_credentials parameter
- `client_id` (uuid4()): client_id parameter
- `client_secret` (str): client_secret parameter
- `tenant_id` (uuid4()): tenant_id parameter

Response Structure:

- `IsLoggedIn` (Optional[bool]): Isloggedin value (single)
- `Environment` (Optional[str]): Environment value (single)
- `UserName` (Optional[str]): Username value (single)
- `TenantName` (Optional[str]): Tenantname value (single)

```
use_client_credentials = True
client_id = "550e8400-e29b-41d4-a716-446655440000"
client_secret = "example_value"
```

```
tenant_id = "550e8400-e29b-41d4-a716-446655440000"

auth_login_client_credentials_resp: list[CommandResponse[Contracts_LoginResponse]]
= pxc.auth.login_client_credentials(use_client_credentials=use_client_credentials,
client_id=client_id, client_secret=client_secret, tenant_id=tenant_id,
print_message=True)
auth_login_client_credentials_final: Contracts_LoginResponse =
SDKBase.get_response_data(auth_login_client_credentials_resp)

if auth_login_client_credentials_final is not None:
    # Access single properties
    if auth_login_client_credentials_final.IsLoggedIn is not None:
        print(f"IsLoggedIn: {auth_login_client_credentials_final.IsLoggedIn}")
    if auth_login_client_credentials_final.Environment is not None:
        print(f"Environment: {auth_login_client_credentials_final.Environment}")
    if auth_login_client_credentials_final.UserName is not None:
        print(f"UserName: {auth_login_client_credentials_final.UserName}")
    if auth_login_client_credentials_final.TenantName is not None:
        print(f"TenantName: {auth_login_client_credentials_final.TenantName}")
else:
    print(f"login_client_credentials failed:
{auth_login_client_credentials_resp.Message}")
```

Login

Description: Authenticate user via interactive login

Response Structure:

- `IsLoggedIn` (Optional[bool]): Isloggedin value (single)
- `Environment` (Optional[str]): Environment value (single)
- `UserName` (Optional[str]): Username value (single)
- `TenantName` (Optional[str]): Tenantname value (single)

```
auth_login_resp: list[CommandResponse[Contracts_LoginResponse]] =
pxc.auth.login(print_message=True)
auth_login_final: Contracts_LoginResponse =
SDKBase.get_response_data(auth_login_resp)

if auth_login_final is not None:
    # Access single properties
    if auth_login_final.IsLoggedIn is not None:
        print(f"IsLoggedIn: {auth_login_final.IsLoggedIn}")
    if auth_login_final.Environment is not None:
        print(f"Environment: {auth_login_final.Environment}")
    if auth_login_final.UserName is not None:
        print(f"UserName: {auth_login_final.UserName}")
    if auth_login_final.TenantName is not None:
        print(f"TenantName: {auth_login_final.TenantName}")
```

```
else:
    print(f"login failed: {auth_login_resp.Message}")
```

Logout

Description: Log out the current user

Response Structure:

- `IsLoggedOut` (Optional[bool]): Isloggedout value (single)

```
auth_logout_resp: list[CommandResponse[Contracts_LoggedOutResponse]] =
pxc.auth.logout(print_message=True)
auth_logout_final: Contracts_LoggedOutResponse =
SDKBase.get_response_data(auth_logout_resp)

if auth_logout_final is not None:
    # Access single properties
    if auth_logout_final.IsLoggedOut is not None:
        print(f"IsLoggedOut: {auth_logout_final.IsLoggedOut}")
else:
    print(f"logout failed: {auth_logout_resp.Message}")
```

Datahub

Delete

Description: Delete files from Datahub

Parameters:

- `remote_glob_patterns` (list[str]): Glob patterns to match remote files

Response Structure:

- `Success` (Optional[bool]): Success value (single)
- `QueuedForDeletion` (Optional[list[str]]): List of Queuedfordeletion (list)

```
remote_glob_patterns = ["folder/**/*.csv", "folder/**/*.parquet"]

datahub_delete_resp: list[CommandResponse[Contracts_DatahubDeleteResponse]] =
pxc.datahub.delete(remote_glob_patterns=remote_glob_patterns, print_message=True)
datahub_delete_final: Contracts_DatahubDeleteResponse =
SDKBase.get_response_data(datahub_delete_resp)

if datahub_delete_final is not None:
    # Iterate over QueuedForDeletion list
    if datahub_delete_final.QueuedForDeletion is not None:
        for item in datahub_delete_final.QueuedForDeletion:
            print(item)
    else:
        print(f"No QueuedForDeletion returned")
    # Access single properties
    if datahub_delete_final.Success is not None:
        print(f"Success: {datahub_delete_final.Success}")
else:
    print(f"delete failed: {datahub_delete_resp.Message}")
```

Download

Description: Download files from Datahub using glob patterns

Parameters:

- `remote_glob_patterns` (list[str]): Glob patterns to match remote files (optional)
- `output_directory` (str): Local directory to save downloaded files (optional, default: None)
- `version` (any): version parameter (optional, default: None)
- `snapshot_date` (any): snapshot_date parameter (optional, default: None)
- `manifest_file_path` (str): manifest_file_path parameter (optional, default: None)
- `verify_download` (bool): Verify downloaded files for integrity (optional, default: None)
- `create_metadata_file` (bool): create_metadata_file parameter (optional, default: None)
- `what_if_verification` (bool): what_if_verification parameter (optional, default: None)

Response Structure:

- `DatahubCommandStatus` (Optional[Datahub_DatahubCommandStatus]): Datahubcommandstatus value (single)

  ▶ Properties of `Datahub_DatahubCommandStatus`
  - `Success` (str): Success value
  - `PartialSuccess` (str): Partialsuccess value
  - `Failure` (str): Failure value

- `DatahubResourceResults` (Optional[list[Datahub_DatahubResourceResult]]): List of Datahubresourceresults (list)

  ▶ Properties of `Datahub_DatahubResourceResult`
  - `RelativeFilePath` (Optional[str]): Relativefilepath value
  - `RelativeToDirectoryOutputPath` (Optional[str]): Relativetodirectoryoutputpath value
  - `LocalFilePath` (Optional[str]): Localfilepath value
  - `FailureReason` (Optional[str]): Failurereason value
  - `Success` (Optional[bool]): Success value
  - `Version` (Optional[int]): Version value
  - `IsFromConnector` (Optional[bool]): Isfromconnector value
  - `SnapshotDateUtc` (Optional[str]): Snapshotdateutc value

```
remote_glob_patterns = ["folder/**/*.csv", "folder/**/*.parquet"]
output_directory = r"c:\output"
version = None
snapshot_date = None
manifest_file_path = r"c:\path\to\manifest.csv"
verify_download = True
create_metadata_file = True
what_if_verification = True

datahub_download_resp: list[CommandResponse[Contracts_DatahubCommandResponse]] =
pxc.datahub.download(remote_glob_patterns=remote_glob_patterns,
output_directory=output_directory, version=version, snapshot_date=snapshot_date,
manifest_file_path=manifest_file_path, verify_download=verify_download,
create_metadata_file=create_metadata_file,
what_if_verification=what_if_verification, print_message=True)
datahub_download_final: Contracts_DatahubCommandResponse =
SDKBase.get_response_data(datahub_download_resp)

if datahub_download_final is not None:
    # Iterate over DatahubResourceResults list
    if datahub_download_final.DatahubResourceResults is not None:
        for item in datahub_download_final.DatahubResourceResults:
            # Access properties of Datahub_DatahubResourceResult object
            print(f"RelativeFilePath: {item.RelativeFilePath}")
            print(f"RelativeToDirectoryOutputPath:
{item.RelativeToDirectoryOutputPath}")
            print(f"LocalFilePath: {item.LocalFilePath}")
            print(f"FailureReason: {item.FailureReason}")
            print(f"Success: {item.Success}")
            # ... and 3 more properties
    else:
        print(f"No DatahubResourceResults returned")
    # Access single properties
    if datahub_download_final.DatahubCommandStatus is not None:
        print(f"DatahubCommandStatus:
{datahub_download_final.DatahubCommandStatus}")
else:
    print(f"download failed: {datahub_download_resp.Message}")
```

Revert

Description: Execute revert operation

Parameters:

- `file_revert_path` (str): file_revert_path parameter
- `version` (any): version parameter (optional)
- `snapshot_date` (any): snapshot_date parameter (optional)
- `verify_download` (bool): Verify downloaded files for integrity (optional, default: None)

Response Structure:

- `DatahubCommandStatus` (Optional[Datahub_DatahubCommandStatus]): Datahubcommandstatus value (single)

  ▶ Properties of `Datahub_DatahubCommandStatus`
  - `Success` (str): Success value
  - `PartialSuccess` (str): Partialsuccess value
  - `Failure` (str): Failure value

- `DatahubResourceResults` (Optional[list[Datahub_DatahubResourceResult]]): List of Datahubresourceresults (list)

  ▶ Properties of `Datahub_DatahubResourceResult`
  - `RelativeFilePath` (Optional[str]): Relativefilepath value
  - `RelativeToDirectoryOutputPath` (Optional[str]): Relativetodirectoryoutputpath value
  - `LocalFilePath` (Optional[str]): Localfilepath value
  - `FailureReason` (Optional[str]): Failurereason value
  - `Success` (Optional[bool]): Success value
  - `Version` (Optional[int]): Version value
  - `IsFromConnector` (Optional[bool]): Isfromconnector value
  - `SnapshotDateUtc` (Optional[str]): Snapshotdateutc value

```
file_revert_path = r"/path/folder/file.csv"
version = None
snapshot_date = None
verify_download = True

datahub_revert_resp: list[CommandResponse[Contracts_DatahubCommandResponse]] =
pxc.datahub.revert(file_revert_path=file_revert_path, version=version,
snapshot_date=snapshot_date, verify_download=verify_download, print_message=True)
datahub_revert_final: Contracts_DatahubCommandResponse =
SDKBase.get_response_data(datahub_revert_resp)

if datahub_revert_final is not None:
    # Iterate over DatahubResourceResults list
    if datahub_revert_final.DatahubResourceResults is not None:
        for item in datahub_revert_final.DatahubResourceResults:
            # Access properties of Datahub_DatahubResourceResult object
            print(f"RelativeFilePath: {item.RelativeFilePath}")
            print(f"RelativeToDirectoryOutputPath:
{item.RelativeToDirectoryOutputPath}")
            print(f"LocalFilePath: {item.LocalFilePath}")
            print(f"FailureReason: {item.FailureReason}")
            print(f"Success: {item.Success}")
            # ... and 3 more properties
    else:
        print(f"No DatahubResourceResults returned")
    # Access single properties
    if datahub_revert_final.DatahubCommandStatus is not None:
        print(f"DatahubCommandStatus:
{datahub_revert_final.DatahubCommandStatus}")
```

```
else:
    print(f"revert failed: {datahub_revert_resp.Message}")
```

Search

Description: Search for files in Datahub

Parameters:

- `glob_patterns` (list[str]): glob_patterns parameter
- `include_deleted_files` (bool): include_deleted_files parameter (optional)

Response Structure:

- `Success` (Optional[bool]): Success value (single)
- `DatahubSearchResults` (Optional[list[Datahub_DatahubResourceInfo]]): List of Datahubsearchresults (list)

  ▶ Properties of `Datahub_DatahubResourceInfo`
  - `RelativePath` (Optional[str]): Relativepath value
  - `CreatedAtUtc` (Optional[str]): Createdatutc value
  - `IsDeleted` (Optional[bool]): Isdeleted value
  - `IsSymlink` (Optional[bool]): Issymlink value
  - `IsVersioned` (Optional[bool]): Isversioned value
  - `IsFromConnector` (Optional[bool]): Isfromconnector value
  - `FileSize` (Optional[int]): Filesize value
  - `DeletedAtUtc` (Optional[str]): Deletedatutc value
  - `LatestServerVersion` (Optional[int]): Latestserverversion value
  - `Versions` (Optional[list[Datahub_DatahubVersionInfo]]): List of Versions

```
glob_patterns = ["folder/**/*.csv", "folder/**/*.parquet"]
include_deleted_files = True

datahub_search_resp: list[CommandResponse[Contracts_DatahubSearchResponse]] =
pxc.datahub.search(glob_patterns=glob_patterns,
include_deleted_files=include_deleted_files, print_message=True)
datahub_search_final: Contracts_DatahubSearchResponse =
SDKBase.get_response_data(datahub_search_resp)

if datahub_search_final is not None:
    # Iterate over DatahubSearchResults list
    if datahub_search_final.DatahubSearchResults is not None:
        for item in datahub_search_final.DatahubSearchResults:
            # Access properties of Datahub_DatahubResourceInfo object
            print(f"RelativePath: {item.RelativePath}")
            print(f"CreatedAtUtc: {item.CreatedAtUtc}")
            print(f"IsDeleted: {item.IsDeleted}")
            print(f"IsSymlink: {item.IsSymlink}")
            print(f"IsVersioned: {item.IsVersioned}")
            # ... and 5 more properties
    else:
        print(f"No DatahubSearchResults returned")
    # Access single properties
    if datahub_search_final.Success is not None:
        print(f"Success: {datahub_search_final.Success}")
else:
    print(f"search failed: {datahub_search_resp.Message}")
```

Sync

Description: Synchronize local files with Datahub

Parameters:

- `sync_all_paths` (bool): sync_all_paths parameter (optional)
- `local_path_to_sync` (str): local_path_to_sync parameter (optional, default: None)
- `verify_downloads` (bool): verify_downloads parameter (optional, default: None)
- `replace_local_files_on_conflict` (bool): replace_local_files_on_conflict parameter (optional, default: None)

Response Structure:

- `DatahubCommandStatus` (Optional[Datahub_DatahubCommandStatus]): Datahubcommandstatus value (single)

  ▶ Properties of `Datahub_DatahubCommandStatus`
    - `Success` (str): Success value
    - `PartialSuccess` (str): Partialsuccess value
    - `Failure` (str): Failure value

- `DatahubResourceResults` (Optional[list[Datahub_DatahubResourceResult]]): List of Datahubresourceresults (list)

  ▶ Properties of `Datahub_DatahubResourceResult`
    - `RelativeFilePath` (Optional[str]): Relativefilepath value
    - `RelativeToDirectoryOutputPath` (Optional[str]): Relativetodirectoryoutputpath value
    - `LocalFilePath` (Optional[str]): Localfilepath value
    - `FailureReason` (Optional[str]): Failurereason value
    - `Success` (Optional[bool]): Success value
    - `Version` (Optional[int]): Version value
    - `IsFromConnector` (Optional[bool]): Isfromconnector value
    - `SnapshotDateUtc` (Optional[str]): Snapshotdateutc value

```
sync_all_paths = True
local_path_to_sync = r"c:\local\path"
verify_downloads = True
replace_local_files_on_conflict = True
```

```
datahub_sync_resp: list[CommandResponse[Contracts_DatahubCommandResponse]] =
pxc.datahub.sync(sync_all_paths=sync_all_paths,
local_path_to_sync=local_path_to_sync, verify_downloads=verify_downloads,
replace_local_files_on_conflict=replace_local_files_on_conflict,
print_message=True)
datahub_sync_final: Contracts_DatahubCommandResponse =
SDKBase.get_response_data(datahub_sync_resp)

if datahub_sync_final is not None:
    # Iterate over DatahubResourceResults list
    if datahub_sync_final.DatahubResourceResults is not None:
        for item in datahub_sync_final.DatahubResourceResults:
            # Access properties of Datahub_DatahubResourceResult object
            print(f"RelativeFilePath: {item.RelativeFilePath}")
            print(f"RelativeToDirectoryOutputPath:
{item.RelativeToDirectoryOutputPath}")
            print(f"LocalFilePath: {item.LocalFilePath}")
            print(f"FailureReason: {item.FailureReason}")
            print(f"Success: {item.Success}")
            # ... and 3 more properties
    else:
        print(f"No DatahubResourceResults returned")
    # Access single properties
    if datahub_sync_final.DatahubCommandStatus is not None:
        print(f"DatahubCommandStatus: {datahub_sync_final.DatahubCommandStatus}")
else:
    print(f"sync failed: {datahub_sync_resp.Message}")
```

Undelete

Description: Execute undelete operation

Parameters:

- `glob_patterns` (list[str]): glob_patterns parameter

Response Structure:

- `Success` (Optional[bool]): Success value (single)
- `QueuedForUndeletion` (Optional[list[str]]): List of Queuedforundeletion (list)

```
glob_patterns = ["folder/**/*.csv", "folder/**/*.parquet"]

datahub_undelete_resp: list[CommandResponse[Contracts_DatahubUnDeleteResponse]] =
pxc.datahub.undelete(glob_patterns=glob_patterns, print_message=True)
datahub_undelete_final: Contracts_DatahubUnDeleteResponse =
SDKBase.get_response_data(datahub_undelete_resp)

if datahub_undelete_final is not None:
    # Iterate over QueuedForUndeletion list
    if datahub_undelete_final.QueuedForUndeletion is not None:
        for item in datahub_undelete_final.QueuedForUndeletion:
            print(item)
    else:
        print(f"No QueuedForUndeletion returned")
    # Access single properties
    if datahub_undelete_final.Success is not None:
        print(f"Success: {datahub_undelete_final.Success}")
else:
    print(f"undelete failed: {datahub_undelete_resp.Message}")
```

Upload

Description: Upload files to Datahub

Parameters:

- `local_folder` (str): local_folder parameter
- `remote_folder` (str): remote_folder parameter
- `glob_patterns` (list[str]): glob_patterns parameter (optional)
- `is_versioned` (bool): is_versioned parameter (optional)

Response Structure:

- `DatahubCommandStatus` (Optional[Datahub_DatahubCommandStatus]): Datahubcommandstatus value (single)

  ▶ Properties of `Datahub_DatahubCommandStatus`
    - `Success` (str): Success value
    - `PartialSuccess` (str): Partialsuccess value
    - `Failure` (str): Failure value

- `DatahubResourceResults` (Optional[list[Datahub_DatahubResourceResult]]): List of Datahubresourceresults (list)

  ▶ Properties of `Datahub_DatahubResourceResult`
    - `RelativeFilePath` (Optional[str]): Relativefilepath value
    - `RelativeToDirectoryOutputPath` (Optional[str]): Relativetodirectoryoutputpath value
    - `LocalFilePath` (Optional[str]): Localfilepath value
    - `FailureReason` (Optional[str]): Failurereason value
    - `Success` (Optional[bool]): Success value
    - `Version` (Optional[int]): Version value
    - `IsFromConnector` (Optional[bool]): Isfromconnector value
    - `SnapshotDateUtc` (Optional[str]): Snapshotdateutc value

```
local_folder = r"c:\local\folder"
remote_folder = "remote/folder"
glob_patterns = ["folder/**/*.csv", "folder/**/*.parquet"]
is_versioned = True

datahub_upload_resp: list[CommandResponse[Contracts_DatahubCommandResponse]] =
pxc.datahub.upload(local_folder=local_folder, remote_folder=remote_folder,
glob_patterns=glob_patterns, is_versioned=is_versioned, print_message=True)
datahub_upload_final: Contracts_DatahubCommandResponse =
SDKBase.get_response_data(datahub_upload_resp)

if datahub_upload_final is not None:
    # Iterate over DatahubResourceResults list
    if datahub_upload_final.DatahubResourceResults is not None:
        for item in datahub_upload_final.DatahubResourceResults:
            # Access properties of Datahub_DatahubResourceResult object
            print(f"RelativeFilePath: {item.RelativeFilePath}")
            print(f"RelativeToDirectoryOutputPath:
{item.RelativeToDirectoryOutputPath}")
            print(f"LocalFilePath: {item.LocalFilePath}")
            print(f"FailureReason: {item.FailureReason}")
            print(f"Success: {item.Success}")
            # ... and 3 more properties
    else:
        print(f"No DatahubResourceResults returned")
    # Access single properties
    if datahub_upload_final.DatahubCommandStatus is not None:
        print(f"DatahubCommandStatus:
{datahub_upload_final.DatahubCommandStatus}")
else:
    print(f"upload failed: {datahub_upload_resp.Message}")
```

Create Local Symlink

Description: Execute create_local_symlink operation

Parameters:

- `display_name` (str): display_name parameter
- `target_remote_path` (str): target_remote_path parameter
- `symlink_path` (str): symlink_path parameter
- `symlink_type` (any): symlink_type parameter

Response Structure:

- `Success` (Optional[bool]): Success value (single)

```
display_name = "My Display Name"
target_remote_path = "remote/target/path"
symlink_path = r"path/folder"
symlink_type = "example_value"

datahub_create_local_symlink_resp:
list[CommandResponse[Contracts_DatahubSymlinkResponse]] =
pxc.datahub.create_local_symlink(display_name=display_name,
target_remote_path=target_remote_path, symlink_path=symlink_path,
symlink_type=symlink_type, print_message=True)
datahub_create_local_symlink_final: Contracts_DatahubSymlinkResponse =
SDKBase.get_response_data(datahub_create_local_symlink_resp)

if datahub_create_local_symlink_final is not None:
    # Access single properties
    if datahub_create_local_symlink_final.Success is not None:
        print(f"Success: {datahub_create_local_symlink_final.Success}")
else:
    print(f"create_local_symlink failed:
{datahub_create_local_symlink_resp.Message}")
```

Create Symlink

Description: Execute create_symlink operation

Parameters:

- `display_name` (str): display_name parameter
- `target_tenant_id` (str): target_tenant_id parameter
- `target_remote_path` (str): target_remote_path parameter
- `symlink_path` (str): symlink_path parameter
- `symlink_type` (any): symlink_type parameter

Response Structure:

- `Success` (Optional[bool]): Success value (single)

```
display_name = "My Display Name"
target_tenant_id = "550e8400-e29b-41d4-a716-446655440000"
target_remote_path = "remote/target/path"
symlink_path = r"path/folder"
symlink_type = "example_value"

datahub_create_symlink_resp:
list[CommandResponse[Contracts_DatahubSymlinkResponse]] =
pxc.datahub.create_symlink(display_name=display_name,
target_tenant_id=target_tenant_id, target_remote_path=target_remote_path,
symlink_path=symlink_path, symlink_type=symlink_type, print_message=True)
datahub_create_symlink_final: Contracts_DatahubSymlinkResponse =
SDKBase.get_response_data(datahub_create_symlink_resp)

if datahub_create_symlink_final is not None:
    # Access single properties
    if datahub_create_symlink_final.Success is not None:
        print(f"Success: {datahub_create_symlink_final.Success}")
else:
    print(f"create_symlink failed: {datahub_create_symlink_resp.Message}")
```

Delete Symlink

Description: Execute delete_symlink operation

Parameters:

- `symlink_path` (str): symlink_path parameter

Response Structure:

- `Success` (Optional[bool]): Success value (single)

```
symlink_path = r"path/folder"

datahub_delete_symlink_resp:
list[CommandResponse[Contracts_DatahubDeleteSymlinkResponse]] =
pxc.datahub.delete_symlink(symlink_path=symlink_path, print_message=True)
datahub_delete_symlink_final: Contracts_DatahubDeleteSymlinkResponse =
SDKBase.get_response_data(datahub_delete_symlink_resp)

if datahub_delete_symlink_final is not None:
    # Access single properties
    if datahub_delete_symlink_final.Success is not None:
        print(f"Success: {datahub_delete_symlink_final.Success}")
else:
    print(f"delete_symlink failed: {datahub_delete_symlink_resp.Message}")
```

List Symlinks

Description: Execute list_symlinks operation

Response Structure:

- `Success` (Optional[bool]): Success value (single)
- `Symlinks` (Optional[list[Datahub_DatahubSymlinkInfo]]): List of Symlinks (list)

  ▶ Properties of `Datahub_DatahubSymlinkInfo`
  - `DisplayName` (Optional[str]): Displayname value
  - `SymlinkId` (Optional[str]): Symlinkid value
  - `Type` (Optional[Datahub_DatahubSymlinkType]): Type value
  - `TargetTenantId` (Optional[str]): Targettenantid value
  - `RemotePath` (Optional[str]): Remotepath value
  - `SymlinkPath` (Optional[str]): Symlinkpath value

```


datahub_list_symlinks_resp:
list[CommandResponse[Contracts_DatahubListSymlinksResponse]] =
pxc.datahub.list_symlinks(print_message=True)
datahub_list_symlinks_final: Contracts_DatahubListSymlinksResponse =
SDKBase.get_response_data(datahub_list_symlinks_resp)

if datahub_list_symlinks_final is not None:
    # Iterate over Symlinks list
```

```python
    if datahub_list_symlinks_final.Symlinks is not None:
        for item in datahub_list_symlinks_final.Symlinks:
            # Access properties of Datahub_DatahubSymlinkInfo object
            print(f"DisplayName: {item.DisplayName}")
            print(f"SymlinkId: {item.SymlinkId}")
            print(f"Type: {item.Type}")
            print(f"TargetTenantId: {item.TargetTenantId}")
            print(f"RemotePath: {item.RemotePath}")
            # ... and 1 more properties
    else:
        print(f"No Symlinks returned")
    # Access single properties
    if datahub_list_symlinks_final.Success is not None:
        print(f"Success: {datahub_list_symlinks_final.Success}")
else:
    print(f"list_symlinks failed: {datahub_list_symlinks_resp.Message}")
```

Create Share

Description: Execute create_share operation

Parameters:

- `display_name` (str): display_name parameter
- `remote_path` (str): remote_path parameter
- `permissions` (list[str]): permissions parameter (optional)
- `permissions_file_path` (str): permissions_file_path parameter (optional)

Response Structure:

- `Success` (Optional[bool]): Success value (single)

```python
display_name = "My Display Name"
remote_path = "remote/path/*.*"
permissions = ["read", "write"]
permissions_file_path = r"c:\path\to\permissions.txt"

datahub_create_share_resp: list[CommandResponse[Contracts_DatahubShareResponse]] =
pxc.datahub.create_share(display_name=display_name, remote_path=remote_path,
permissions=permissions, permissions_file_path=permissions_file_path,
print_message=True)
datahub_create_share_final: Contracts_DatahubShareResponse =
SDKBase.get_response_data(datahub_create_share_resp)

if datahub_create_share_final is not None:
    # Access single properties
    if datahub_create_share_final.Success is not None:
        print(f"Success: {datahub_create_share_final.Success}")
else:
    print(f"create_share failed: {datahub_create_share_resp.Message}")
```

Delete Share

Description: Execute delete_share operation

Parameters:

- `share_id` (uuid4()): share_id parameter

Response Structure:

- `Success` (Optional[bool]): Success value (single)

```
share_id = "550e8400-e29b-41d4-a716-446655440000"

datahub_delete_share_resp:
list[CommandResponse[Contracts_DatahubDeleteShareResponse]] =
pxc.datahub.delete_share(share_id=share_id, print_message=True)
datahub_delete_share_final: Contracts_DatahubDeleteShareResponse =
SDKBase.get_response_data(datahub_delete_share_resp)

if datahub_delete_share_final is not None:
    # Access single properties
    if datahub_delete_share_final.Success is not None:
        print(f"Success: {datahub_delete_share_final.Success}")
else:
    print(f"delete_share failed: {datahub_delete_share_resp.Message}")
```

List Shares

Description: Execute list_shares operation

Response Structure:

- `Success` (Optional[bool]): Success value (single)
- `Shares` (Optional[list[Datahub_DatahubShareInfo]]): List of Shares (list)

▶ Properties of `Datahub_DatahubShareInfo`

- `ShareId` (Optional[str]): Shareid value
- `Name` (Optional[str]): Name value
- `RelativePath` (Optional[str]): Relativepath value
- `Permissions` (Optional[list[Datahub_DatahubSharePermissionInfo]]): List of Permissions

```
datahub_list_shares_resp:
list[CommandResponse[Contracts_DatahubListSharesResponse]] =
pxc.datahub.list_shares(print_message=True)
datahub_list_shares_final: Contracts_DatahubListSharesResponse =
SDKBase.get_response_data(datahub_list_shares_resp)
```

```
if datahub_list_shares_final is not None:
    # Iterate over Shares list
    if datahub_list_shares_final.Shares is not None:
        for item in datahub_list_shares_final.Shares:
            # Access properties of Datahub_DatahubShareInfo object
            print(f"ShareId: {item.ShareId}")
            print(f"Name: {item.Name}")
            print(f"RelativePath: {item.RelativePath}")
            print(f"Permissions: {item.Permissions}")
    else:
        print(f"No Shares returned")
    # Access single properties
    if datahub_list_shares_final.Success is not None:
        print(f"Success: {datahub_list_shares_final.Success}")
else:
    print(f"list_shares failed: {datahub_list_shares_resp.Message}")
```

Add Or Update Permission

Description: Execute add_or_update_permission operation

Parameters:

- `remote_path` (str): remote_path parameter
- `type` (any): type parameter
- `user_or_role_id` (uuid4()): user_or_role_id parameter
- `read` (bool): read parameter
- `write` (bool): write parameter
- `inherit_parent_permissions` (bool): inherit_parent_permissions parameter (optional)

Response Structure:

- `success` (Optional[bool]): Success value (single)

```
remote_path = "remote/path/*.*"
type = "example_value"
user_or_role_id = "550e8400-e29b-41d4-a716-446655440000"
read = True
write = True
inherit_parent_permissions = True

datahub_add_or_update_permission_resp:
list[CommandResponse[Contracts_DatahubAddOrUpdatePermissionResponse]] =
pxc.datahub.add_or_update_permission(remote_path=remote_path, type=type,
user_or_role_id=user_or_role_id, read=read, write=write,
inherit_parent_permissions=inherit_parent_permissions, print_message=True)
datahub_add_or_update_permission_final:
Contracts_DatahubAddOrUpdatePermissionResponse =
SDKBase.get_response_data(datahub_add_or_update_permission_resp)

if datahub_add_or_update_permission_final is not None:
    # Access single properties
    if datahub_add_or_update_permission_final.success is not None:
        print(f"success: {datahub_add_or_update_permission_final.success}")
else:
    print(f"add_or_update_permission failed: 
{datahub_add_or_update_permission_resp.Message}")
```

Delete Permission

Description: Execute delete_permission operation

Parameters:

- `remote_path` (str): remote_path parameter
- `roles` (list[str]): roles parameter
- `user_ids` (list[str]): user_ids parameter

Response Structure:

- `success` (Optional[bool]): Success value (single)

```
remote_path = "remote/path/*.*"
roles = ["admin", "user"]
user_ids = ["user1", "user2"]

datahub_delete_permission_resp: 
list[CommandResponse[Contracts_DatahubDeletePermissionResponse]] = 
pxc.datahub.delete_permission(remote_path=remote_path, roles=roles, 
user_ids=user_ids, print_message=True)
datahub_delete_permission_final: Contracts_DatahubDeletePermissionResponse = 
SDKBase.get_response_data(datahub_delete_permission_resp)

if datahub_delete_permission_final is not None:
    # Access single properties
    if datahub_delete_permission_final.success is not None:
        print(f"success: {datahub_delete_permission_final.success}")
else:
    print(f"delete_permission failed: {datahub_delete_permission_resp.Message}")
```

Delete Permission Rule

Description: Execute delete_permission_rule operation

Parameters:

- `remote_path` (str): remote_path parameter

Response Structure:

- `success` (Optional[bool]): Success value (single)

```
remote_path = "remote/path/*.*"

datahub_delete_permission_rule_resp:
list[CommandResponse[Contracts_DatahubDeleteRuleResponse]] =
pxc.datahub.delete_permission_rule(remote_path=remote_path, print_message=True)
datahub_delete_permission_rule_final: Contracts_DatahubDeleteRuleResponse =
SDKBase.get_response_data(datahub_delete_permission_rule_resp)

if datahub_delete_permission_rule_final is not None:
    # Access single properties
    if datahub_delete_permission_rule_final.success is not None:
        print(f"success: {datahub_delete_permission_rule_final.success}")
else:
    print(f"delete_permission_rule failed:
{datahub_delete_permission_rule_resp.Message}")
```

List Permissions

Description: Execute list_permissions operation

Response Structure:

- `success` (Optional[bool]): Success value (single)
- `acls` (Optional[list[Datahub_DatahubAclInfo]]): List of Acls (list)

  ▶ Properties of `Datahub_DatahubAclInfo`
    - `Id` (Optional[str]): Id value
    - `InheritParent` (Optional[bool]): Inheritparent value
    - `RelativePath` (Optional[str]): Relativepath value
    - `Permissions` (Optional[Datahub_UserRolePermissionInfo]): Permissions value

```


datahub_list_permissions_resp:
list[CommandResponse[Contracts_DatahubAclListResponse]] =
pxc.datahub.list_permissions(print_message=True)
datahub_list_permissions_final: Contracts_DatahubAclListResponse =
SDKBase.get_response_data(datahub_list_permissions_resp)

if datahub_list_permissions_final is not None:
    # Iterate over acls list
    if datahub_list_permissions_final.acls is not None:
        for item in datahub_list_permissions_final.acls:
            # Access properties of Datahub_DatahubAclInfo object
            print(f"Id: {item.Id}")
            print(f"InheritParent: {item.InheritParent}")
            print(f"RelativePath: {item.RelativePath}")
            print(f"Permissions: {item.Permissions}")
    else:
        print(f"No acls returned")
    # Access single properties
    if datahub_list_permissions_final.success is not None:
        print(f"success: {datahub_list_permissions_final.success}")
else:
    print(f"list_permissions failed: {datahub_list_permissions_resp.Message}")
```

Update Inherit Permission

Description: Execute update_inherit_permission operation

Parameters:

- `remote_path` (str): remote_path parameter
- `inherit_parent_permissions` (bool): inherit_parent_permissions parameter

Response Structure:

- `success` (Optional[bool]): Success value (single)

```
remote_path = "remote/path/*.*"
inherit_parent_permissions = True

datahub_update_inherit_permission_resp:
list[CommandResponse[Contracts_DatahubUpdateInheritPermissionResponse]] =
pxc.datahub.update_inherit_permission(remote_path=remote_path,
inherit_parent_permissions=inherit_parent_permissions, print_message=True)
datahub_update_inherit_permission_final:
Contracts_DatahubUpdateInheritPermissionResponse =
SDKBase.get_response_data(datahub_update_inherit_permission_resp)

if datahub_update_inherit_permission_final is not None:
    # Access single properties
    if datahub_update_inherit_permission_final.success is not None:
        print(f"success: {datahub_update_inherit_permission_final.success}")
else:
    print(f"update_inherit_permission failed:
{datahub_update_inherit_permission_resp.Message}")
```

Map Folder

Description: Execute map_folder operation

Parameters:

- `local_folder` (str): local_folder parameter
- `remote_folder` (str): remote_folder parameter
- `remote_glob_patterns` (list[str]): Glob patterns to match remote files (optional)

Response Structure:

- `Success` (Optional[bool]): Success value (single)
- `LocalPath` (Optional[str]): Localpath value (single)
- `RemotePath` (Optional[str]): Remotepath value (single)
- `Patterns` (Optional[list[str]]): List of Patterns (list)

```
local_folder = r"c:\local\folder"
remote_folder = "remote/folder"
remote_glob_patterns = ["folder/**/*.csv", "folder/**/*.parquet"]

datahub_map_folder_resp: list[CommandResponse[Contracts_DatahubMapResponse]] =
pxc.datahub.map_folder(local_folder=local_folder, remote_folder=remote_folder,
remote_glob_patterns=remote_glob_patterns, print_message=True)
datahub_map_folder_final: Contracts_DatahubMapResponse =
SDKBase.get_response_data(datahub_map_folder_resp)

if datahub_map_folder_final is not None:
    # Iterate over Patterns list
    if datahub_map_folder_final.Patterns is not None:
        for item in datahub_map_folder_final.Patterns:
            print(item)
    else:
        print(f"No Patterns returned")
    # Access single properties
    if datahub_map_folder_final.Success is not None:
        print(f"Success: {datahub_map_folder_final.Success}")
    if datahub_map_folder_final.LocalPath is not None:
        print(f"LocalPath: {datahub_map_folder_final.LocalPath}")
    if datahub_map_folder_final.RemotePath is not None:
        print(f"RemotePath: {datahub_map_folder_final.RemotePath}")
else:
    print(f"map_folder failed: {datahub_map_folder_resp.Message}")
```

Unmap Folder

Description: Execute unmap_folder operation

Parameters:

- `local_folder_path` (str): local_folder_path parameter

Response Structure:

- `Success` (Optional[bool]): Success value (single)
- `LocalPath` (Optional[str]): Localpath value (single)

```
local_folder_path = "example_value"

datahub_unmap_folder_resp: list[CommandResponse[Contracts_DatahubUnmapResponse]] =
pxc.datahub.unmap_folder(local_folder_path=local_folder_path, print_message=True)
datahub_unmap_folder_final: Contracts_DatahubUnmapResponse =
SDKBase.get_response_data(datahub_unmap_folder_resp)

if datahub_unmap_folder_final is not None:
    # Access single properties
    if datahub_unmap_folder_final.Success is not None:
        print(f"Success: {datahub_unmap_folder_final.Success}")
    if datahub_unmap_folder_final.LocalPath is not None:
        print(f"LocalPath: {datahub_unmap_folder_final.LocalPath}")
else:
    print(f"unmap_folder failed: {datahub_unmap_folder_resp.Message}")
```

Inputdata

Convert Database To Xml

Description: Execute convert_database_to_xml operation

Parameters:

- `db_file_path` (str): db_file_path parameter
- `xml_file_path` (str): xml_file_path parameter
- `study_id` (uuid4()): Unique identifier for a specific study

Response Structure: None

```
db_file_path = r"c:\path\to\database.db"
xml_file_path = r"c:\path\to\data.xml"
study_id = "550e8400-e29b-41d4-a716-446655440000"

inputdata_convert_database_to_xml_resp:
list[CommandResponse[Contracts_ConvertDatabaseToXmlResponse]] =
pxc.inputdata.convert_database_to_xml(db_file_path=db_file_path,
xml_file_path=xml_file_path, study_id=study_id, print_message=True)
inputdata_convert_database_to_xml_final: Contracts_ConvertDatabaseToXmlResponse =
SDKBase.get_response_data(inputdata_convert_database_to_xml_resp)

if inputdata_convert_database_to_xml_final is not None:
    print(f"convert_database_to_xml completed successfully")
else:
    print(f"convert_database_to_xml failed:
{inputdata_convert_database_to_xml_resp.Message}")
```

Convert Xml To Database

Description: Execute convert_xml_to_database operation

Parameters:

- `xml_file_path` (str): xml_file_path parameter
- `db_file_path` (str): db_file_path parameter

- `study_id` (uuid4()): Unique identifier for a specific study

Response Structure: None

```
xml_file_path = r"c:\path\to\data.xml"
db_file_path = r"c:\path\to\database.db"
study_id = "550e8400-e29b-41d4-a716-446655440000"

inputdata_convert_xml_to_database_resp:
list[CommandResponse[Contracts_ConvertXmlToDatabaseResponse]] =
pxc.inputdata.convert_xml_to_database(xml_file_path=xml_file_path,
db_file_path=db_file_path, study_id=study_id, print_message=True)
inputdata_convert_xml_to_database_final: Contracts_ConvertXmlToDatabaseResponse =
SDKBase.get_response_data(inputdata_convert_xml_to_database_resp)

if inputdata_convert_xml_to_database_final is not None:
    print(f"convert_xml_to_database completed successfully")
else:
    print(f"convert_xml_to_database failed:
{inputdata_convert_xml_to_database_resp.Message}")
```

Update Input Data From Json

Description: Execute update_input_data_from_json operation

Parameters:

- `db_file_path` (str): db_file_path parameter
- `json_file_path` (str): json_file_path parameter
- `study_id` (uuid4()): Unique identifier for a specific study

Response Structure:

- `crudBaseData` (Optional[list[Contracts_CrudBaseData]]): List of Crudbasedata (list)

  ▶ Properties of `Contracts_CrudBaseData`
  - `Placeholders` (Optional[dict[str, str]]): Placeholders value
  - `Type` (Optional[InputData_DataType]): Type value
  - `Errors` (Optional[list[str]]): List of Errors

```
db_file_path = r"c:\path\to\database.db"
json_file_path = r"c:\path\to\data.json"
study_id = "550e8400-e29b-41d4-a716-446655440000"

inputdata_update_input_data_from_json_resp:
list[CommandResponse[Contracts_UpdateInputDataFromJsonResponse]] =
pxc.inputdata.update_input_data_from_json(db_file_path=db_file_path,
json_file_path=json_file_path, study_id=study_id, print_message=True)
inputdata_update_input_data_from_json_final:
Contracts_UpdateInputDataFromJsonResponse =
SDKBase.get_response_data(inputdata_update_input_data_from_json_resp)

if inputdata_update_input_data_from_json_final is not None:
    # Iterate over crudBaseData list
    if inputdata_update_input_data_from_json_final.crudBaseData is not None:
        for item in inputdata_update_input_data_from_json_final.crudBaseData:
            # Access properties of Contracts_CrudBaseData object
            print(f"Placeholders: {item.Placeholders}")
            print(f"Type: {item.Type}")
            print(f"Errors: {item.Errors}")
    else:
        print(f"No crudBaseData returned")
else:
    print(f"update_input_data_from_json failed:
{inputdata_update_input_data_from_json_resp.Message}")
```

Log

Parse Log

Description: Execute parse_log operation

Parameters:

- `log_file_path` (str): log_file_path parameter
- `system_object_name` (str): system_object_name parameter (optional)
- `user_locale` (str): user_locale parameter (optional)

Response Structure:

- `LogStepDataList` (Optional[list[Simulation_LogStepDataDto]]): List of Logstepdatalist (list)

  ▶ Properties of `Simulation_LogStepDataDto`
    - `Phase` (Optional[str]): Phase value
    - `Step` (Optional[int]): Step value
    - `Steps` (Optional[int]): Steps value
    - `FromDate` (Optional[str]): Fromdate value
    - `ToDate` (Optional[str]): Todate value
    - `Time` (Optional[System_TimeSpan]): Time value
    - `Elapsed` (Optional[System_TimeSpan]): Elapsed value
    - `Left` (Optional[System_TimeSpan]): Left value
    - `Memory` (Optional[float]): Memory value
    - `Load` (Optional[float]): Load value
    - `Generation` (Optional[float]): Generation value
    - `NetExport` (Optional[float]): Netexport value
    - `GenCost` (Optional[float]): Gencost value
    - `LoadCost` (Optional[float]): Loadcost value
    - `Unserved` (Optional[float]): Unserved value
    - `Price` (Optional[float]): Price value
    - `Solver` (Optional[str]): Solver value

    - `Model` (Optional[str]): Model value
    - `GasDemand` (Optional[float]): Gasdemand value
    - `GasSupply` (Optional[float]): Gassupply value
    - `GasNetExchange` (Optional[float]): Gasnetexchange value
    - `GasDemandCost` (Optional[float]): Gasdemandcost value
    - `GasPrice` (Optional[float]): Gasprice value
    - `GasExcess` (Optional[float]): Gasexcess value
    - `GasShortage` (Optional[float]): Gasshortage value
    - `SummaryData` (Optional[list[Simulation_LogSummaryData]]): List of Summarydata
    - `IterationData` (Optional[list[Simulation_LogIterationData]]): List of Iterationdata

```
log_file_path = r"c:\path\to\PLEXOS_log.txt
system_object_name = "System"
user_locale = "en-US"

log_parse_log_resp: list[CommandResponse[Contracts_ParseLogResponse]] =
pxc.log.parse_log(log_file_path=log_file_path,
system_object_name=system_object_name, user_locale=user_locale,
print_message=True)
log_parse_log_final: Contracts_ParseLogResponse =
SDKBase.get_response_data(log_parse_log_resp)

if log_parse_log_final is not None:
    # Iterate over LogStepDataList list
    if log_parse_log_final.LogStepDataList is not None:
        for item in log_parse_log_final.LogStepDataList:
            # Access properties of Simulation_LogStepDataDto object
            print(f"Phase: {item.Phase}")
            print(f"Step: {item.Step}")
            print(f"Steps: {item.Steps}")
            print(f"FromDate: {item.FromDate}")
            print(f"ToDate: {item.ToDate}")
            # ... and 22 more properties
    else:
        print(f"No LogStepDataList returned")
else:
    print(f"parse_log failed: {log_parse_log_resp.Message}")
```

Simulation

Build Simulation Request From Id

Description: Execute build_simulation_request_from_id operation

Parameters:

- `simulation_id` (uuid4()): Unique identifier for a specific simulation
- `output_directory` (str): Local directory to save downloaded files
- `file_name` (str): file_name parameter

- `overwrite` (bool): overwrite parameter
- `study_id` (uuid4()): Unique identifier for a specific study (optional)
- `changeset_id` (uuid4()): Unique identifier for a specific changeset (optional)
- `model_name` (str): model_name parameter (optional)
- `requested_cpu_cores` (int): requested_cpu_cores parameter (optional)
- `requested_memory` (float): requested_memory parameter (optional)

Response Structure:

- `SimulationContract` (Optional[Simulation_EnqueueSimulationRequest]): Simulationcontract value (single)

  ▶ Properties of `Simulation_EnqueueSimulationRequest`
  - `ExecutionIndex` (Optional[int]): Executionindex value
  - `ExecutionId` (Optional[str]): Executionid value
  - `TenantId` (Optional[str]): Tenantid value
  - `CreatedByUser` (Optional[Simulation_User]): Createdbyuser value
  - `CreatedByUserId` (Optional[str]): Createdbyuserid value
  - `StudyId` (Optional[str]): Studyid value
  - `ChangeSetId` (Optional[str]): Changesetid value
  - `Models` (Optional[list[str]]): List of Models
  - `SimulationOptions` (Optional[Simulation_SimulationOption]): Simulationoptions value
  - `SimulationAffinity` (Optional[Simulation_SimulationAffinity]): Simulationaffinity value
  - `ParallelizationOptions` (Optional[Simulation_ParallelizationOption]): Parallelizationoptions value
  - `SimulationData` (Optional[list[Simulation_SimulationDataUri]]): List of Simulationdata
  - `SolutionData` (Optional[list[Simulation_SolutionData]]): List of Solutiondata
  - `DataConfiguration` (Optional[Simulation_DataConfiguration]): Dataconfiguration value
  - `SimulationEngine` (Optional[Simulation_SimulationEngine]): Simulationengine value
  - `Tags` (Optional[dict[str, str]]): Tags value
  - `Source` (Optional[str]): Source value
  - `Priority` (Optional[int]): Priority value
  - `RequestedCpuCores` (Optional[int]): Requestedcpucores value
  - `MinimumMemoryInGb` (Optional[float]): Minimummemoryingb value
  - `QueueId` (Optional[str]): Queueid value
  - `RunParameters` (Optional[list[Simulation_RunParameters]]): List of Runparameters
  - `RequestedInstanceType` (Optional[Simulation_WorkerPoolInstance]): Requestedinstancetype value
  - `SimulationType` (Optional[Simulation_SimulationTypeEnum]): Simulationtype value

- `fileOutput` (Optional[str]): Fileoutput value (single)

```
simulation_id = "550e8400-e29b-41d4-a716-446655440000"
output_directory = r"c:\output"
file_name = "output.txt"
overwrite = True
study_id = None
changeset_id = None
```

```
model_name = "MyModel"
requested_cpu_cores = 4
requested_memory = 0.0

simulation_build_simulation_request_from_id_resp:
list[CommandResponse[Contracts_BuildSimulationRequestFromIdResponse]] =
pxc.simulation.build_simulation_request_from_id(simulation_id=simulation_id,
output_directory=output_directory, file_name=file_name, overwrite=overwrite,
study_id=study_id, changeset_id=changeset_id, model_name=model_name,
requested_cpu_cores=requested_cpu_cores, requested_memory=requested_memory,
print_message=True)
simulation_build_simulation_request_from_id_final:
Contracts_BuildSimulationRequestFromIdResponse =
SDKBase.get_response_data(simulation_build_simulation_request_from_id_resp)

if simulation_build_simulation_request_from_id_final is not None:
    # Access single properties
    if simulation_build_simulation_request_from_id_final.SimulationContract is not
None:
        print(f"SimulationContract:
{simulation_build_simulation_request_from_id_final.SimulationContract}")
    if simulation_build_simulation_request_from_id_final.fileOutput is not None:
        print(f"fileOutput:
{simulation_build_simulation_request_from_id_final.fileOutput}")
else:
    print(f"build_simulation_request_from_id failed:
{simulation_build_simulation_request_from_id_resp.Message}")
```

Cancel Simulation

Description: Cancel a running simulation

Parameters:

- `simulation_id` (uuid4()): Unique identifier for a specific simulation

Response Structure:

- `SimulationId` (Optional[str]): Simulationid value (single)

- `SimulationCancellationStatus` (Optional[Simulation_SimulationCancellationStatusEnum]): Simulationcancellationstatus value (single)

  ▶ Properties of `Simulation_SimulationCancellationStatusEnum`
    - `Failed` (str): Failed value
    - `Cancelled` (str): Cancelled value
    - `Pending` (str): Pending value

```
simulation_id = "550e8400-e29b-41d4-a716-446655440000"

simulation_cancel_simulation_resp:
```

```
list[CommandResponse[Contracts_CancelSimulationResponse]] =
pxc.simulation.cancel_simulation(simulation_id=simulation_id, print_message=True)
simulation_cancel_simulation_final: Contracts_CancelSimulationResponse =
SDKBase.get_response_data(simulation_cancel_simulation_resp)

if simulation_cancel_simulation_final is not None:
    # Access single properties
    if simulation_cancel_simulation_final.SimulationId is not None:
        print(f"SimulationId: {simulation_cancel_simulation_final.SimulationId}")
    if simulation_cancel_simulation_final.SimulationCancellationStatus is not
None:
        print(f"SimulationCancellationStatus:
{simulation_cancel_simulation_final.SimulationCancellationStatus}")
else:
    print(f"cancel_simulation failed:
{simulation_cancel_simulation_resp.Message}")
```

Check Simulation Progress

Description: Check the progress of a specific simulation

Parameters:

- `simulation_id` (uuid4()): Unique identifier for a specific simulation

Response Structure:

- `SimulationId` (Optional[str]): Simulationid value (single)
- `StudyId` (Optional[str]): Studyid value (single)
- `Status` (Optional[str]): Status value (single)
- `Phase` (Optional[str]): Phase value (single)
- `Value` (Optional[float]): Value value (single)
- `ModelName` (Optional[str]): Modelname value (single)
- `ModelIndex` (Optional[int]): Modelindex value (single)
- `ModelCount` (Optional[int]): Modelcount value (single)
- `Message` (Optional[str]): Message value (single)
- `LastUpdateDate` (Optional[str]): Lastupdatedate value (single)
- `EstimatedTimeToCompletion` (Optional[System_TimeSpan]): Estimatedtimetocompletion value (single)

```
simulation_id = "550e8400-e29b-41d4-a716-446655440000"

simulation_check_simulation_progress_resp:
list[CommandResponse[Contracts_CheckSimulationProgressResponse]] =
pxc.simulation.check_simulation_progress(simulation_id=simulation_id,
print_message=True)
simulation_check_simulation_progress_final:
Contracts_CheckSimulationProgressResponse =
SDKBase.get_response_data(simulation_check_simulation_progress_resp)

if simulation_check_simulation_progress_final is not None:
    # Access single properties
    if simulation_check_simulation_progress_final.SimulationId is not None:
        print(f"SimulationId: 
{simulation_check_simulation_progress_final.SimulationId}")
    if simulation_check_simulation_progress_final.StudyId is not None:
        print(f"StudyId: {simulation_check_simulation_progress_final.StudyId}")
    if simulation_check_simulation_progress_final.Status is not None:
        print(f"Status: {simulation_check_simulation_progress_final.Status}")
    if simulation_check_simulation_progress_final.Phase is not None:
        print(f"Phase: {simulation_check_simulation_progress_final.Phase}")
    if simulation_check_simulation_progress_final.Value is not None:
        print(f"Value: {simulation_check_simulation_progress_final.Value}")
    if simulation_check_simulation_progress_final.ModelName is not None:
        print(f"ModelName: 
{simulation_check_simulation_progress_final.ModelName}")
    if simulation_check_simulation_progress_final.ModelIndex is not None:
        print(f"ModelIndex: 
{simulation_check_simulation_progress_final.ModelIndex}")
    if simulation_check_simulation_progress_final.ModelCount is not None:
        print(f"ModelCount: 
{simulation_check_simulation_progress_final.ModelCount}")
    if simulation_check_simulation_progress_final.Message is not None:
        print(f"Message: {simulation_check_simulation_progress_final.Message}")
    if simulation_check_simulation_progress_final.LastUpdateDate is not None:
        print(f"LastUpdateDate: 
{simulation_check_simulation_progress_final.LastUpdateDate}")
    if simulation_check_simulation_progress_final.EstimatedTimeToCompletion is not 
None:
        print(f"EstimatedTimeToCompletion: 
{simulation_check_simulation_progress_final.EstimatedTimeToCompletion}")
else:
    print(f"check_simulation_progress failed: 
{simulation_check_simulation_progress_resp.Message}")
```

List Simulations

Description: List simulations with optional filtering by ID parameters

Parameters:

- `simulation_id` (uuid4()): Unique identifier for a specific simulation (optional)
- `study_id` (uuid4()): Unique identifier for a specific study (optional, default: None)
- `execution_id` (uuid4()): Unique identifier for a specific execution (optional, default: None)
- `changeset_id` (uuid4()): Unique identifier for a specific changeset (optional, default: None)
- `order_by` (str): Field to order results by (optional, default: None)
- `descending` (bool): Order results in descending order (optional, default: None)
- `top` (int): Maximum number of results to return (optional, default: None)
- `skip` (int): Number of results to skip (optional, default: None)
- `raw` (str): Raw filter string for advanced queries (optional, default: None)

Response Structure:

- `SimulationRecords` (Optional[list[Contracts_Simulation]]): List of Simulationrecords (list)

  ▶ Properties of `Contracts_Simulation`
    - `SimulationType` (Optional[str]): Simulationtype value
    - `Id` (Optional[GuidValue]): Id value
    - `ExecutionId` (Optional[GuidValue]): Executionid value
    - `StudyId` (Optional[GuidValue]): Studyid value
    - `ChangeSetId` (Optional[GuidValue]): Changesetid value
    - `CreatedByUser` (Optional[Contracts_User]): Createdbyuser value
    - `Source` (Optional[str]): Source value
    - `RequestedCpuCores` (Optional[int]): Requestedcpucores value
    - `MinimumMemoryInGb` (Optional[float]): Minimummemoryingb value
    - `CreatedAt` (Optional[str]): Createdat value
    - `LastUpdatedAt` (Optional[str]): Lastupdatedat value
    - `Models` (Optional[list[str]]): List of Models
    - `Status` (Optional[str]): Status value
    - `ModelIdentifiers` (Optional[list[Contracts_ModelIdentifier]]): List of Modelidentifiers
    - `SimulationEngine` (Optional[Contracts_SimulationEngine]): Simulationengine value
    - `RetryCount` (Optional[int]): Retrycount value
    - `Retries` (Optional[list[Contracts_RetriedSimulation]]): List of Retries
    - `ParallelizationOptions` (Optional[Contracts_ParallelizationOptions]): Parallelizationoptions value
    - `Messages` (Optional[list[Contracts_SimulationMessage]]): List of Messages

```python
simulation_id = None
study_id = None
execution_id = None
changeset_id = None
order_by = "CreatedAt"
descending = True
top = 10
skip = 0
raw = "filter expression"

simulation_list_simulations_resp:
list[CommandResponse[Contracts_ListSimulationResponse]] =
pxc.simulation.list_simulations(simulation_id=simulation_id, study_id=study_id,
execution_id=execution_id, changeset_id=changeset_id, order_by=order_by,
descending=descending, top=top, skip=skip, raw=raw, print_message=True)
simulation_list_simulations_final: Contracts_ListSimulationResponse =
SDKBase.get_response_data(simulation_list_simulations_resp)

if simulation_list_simulations_final is not None:
    # Iterate over SimulationRecords list
    if simulation_list_simulations_final.SimulationRecords is not None:
        for item in simulation_list_simulations_final.SimulationRecords:
            # Access properties of Contracts_Simulation object
            print(f"SimulationType: {item.SimulationType}")
            print(f"Id: {item.Id}")
            print(f"ExecutionId: {item.ExecutionId}")
            print(f"StudyId: {item.StudyId}")
            print(f"ChangeSetId: {item.ChangeSetId}")
            # ... and 14 more properties
    else:
        print(f"No SimulationRecords returned")
else:
    print(f"list_simulations failed: {simulation_list_simulations_resp.Message}")
```

Monitor Simulation Progress

Description: Execute monitor_simulation_progress operation

Parameters:

- `simulation_id` (uuid4()): Unique identifier for a specific simulation
- `output` (str): output parameter (optional)

Response Structure:

- `UtilizationData` (Optional[list[Simulation_AgentResourceUtilizationDataV2]]): List of Utilizationdata (list)

  ▶ Properties of `Simulation_AgentResourceUtilizationDataV2`
    - `TimeStamp` (Optional[str]): Timestamp value
    - `FreeMemory` (Optional[float]): Freememory value
    - `UsedMemory` (Optional[float]): Usedmemory value
    - `CPU` (Optional[float]): Cpu value
    - `MemoryPercent` (Optional[float]): Memorypercent value
    - `SwapUsed` (Optional[float]): Swapused value
    - `DriveFreeSpace` (Optional[list[Simulation_HostDriveMetricsV2]]): List of Drivefreespace
    - `WorkingDirectoryDriveMetrics` (Optional[Simulation_HostDriveMetricsV2]): Workingdirectorydrivemetrics value

```
simulation_id = "550e8400-e29b-41d4-a716-446655440000"
output = "detailed"

simulation_monitor_simulation_progress_resp:
list[CommandResponse[Contracts_SimulationAgentResourceUtilizationResponse]] =
pxc.simulation.monitor_simulation_progress(simulation_id=simulation_id,
output=output, print_message=True)
simulation_monitor_simulation_progress_final:
Contracts_SimulationAgentResourceUtilizationResponse =
SDKBase.get_response_data(simulation_monitor_simulation_progress_resp)

if simulation_monitor_simulation_progress_final is not None:
    # Iterate over UtilizationData list
    if simulation_monitor_simulation_progress_final.UtilizationData is not None:
        for item in simulation_monitor_simulation_progress_final.UtilizationData:
            # Access properties of Simulation_AgentResourceUtilizationDataV2
```

```
object
            print(f"TimeStamp: {item.TimeStamp}")
            print(f"FreeMemory: {item.FreeMemory}")
            print(f"UsedMemory: {item.UsedMemory}")
            print(f"CPU: {item.CPU}")
            print(f"MemoryPercent: {item.MemoryPercent}")
            # ... and 3 more properties
    else:
        print(f"No UtilizationData returned")
else:
    print(f"monitor_simulation_progress failed:
{simulation_monitor_simulation_progress_resp.Message}")
```

Run Simulation Group

Description: Execute run_simulation_group operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study
- `simulation_group_id` (uuid4()): simulation_group_id parameter

Response Structure:

- `Data` (Optional[list[Simulation_EnqueuedSimulation]]): List of Data (list)

  ▶ Properties of `Simulation_EnqueuedSimulation`
    - `Id` (Optional[str]): Id value
    - `CreatedAt` (Optional[str]): Createdat value
    - `Status` (Optional[str]): Status value
    - `ExecutionId` (Optional[str]): Executionid value

```
study_id = "550e8400-e29b-41d4-a716-446655440000"
simulation_group_id = "550e8400-e29b-41d4-a716-446655440000"

simulation_run_simulation_group_resp:
list[CommandResponse[Contracts_RunSimulationGroupResponse]] =
pxc.simulation.run_simulation_group(study_id=study_id,
simulation_group_id=simulation_group_id, print_message=True)
simulation_run_simulation_group_final: Contracts_RunSimulationGroupResponse =
SDKBase.get_response_data(simulation_run_simulation_group_resp)

if simulation_run_simulation_group_final is not None:
    # Iterate over Data list
    if simulation_run_simulation_group_final.Data is not None:
        for item in simulation_run_simulation_group_final.Data:
            # Access properties of Simulation_EnqueuedSimulation object
            print(f"Id: {item.Id}")
            print(f"CreatedAt: {item.CreatedAt}")
            print(f"Status: {item.Status}")
            print(f"ExecutionId: {item.ExecutionId}")
    else:
        print(f"No Data returned")
else:
    print(f"run_simulation_group failed:
{simulation_run_simulation_group_resp.Message}")
```

Enqueue Simulation

Description: Enqueue a new simulation for execution

Parameters:

- `file_path` (str): file_path parameter

Response Structure:

- `EnqueueSimulationRequest` (Optional[Simulation_EnqueueSimulationRequest]): Enqueuesimulationrequest value (single)

  ▶ Properties of `Simulation_EnqueueSimulationRequest`
  - `ExecutionIndex` (Optional[int]): Executionindex value
  - `ExecutionId` (Optional[str]): Executionid value
  - `TenantId` (Optional[str]): Tenantid value
  - `CreatedByUser` (Optional[Simulation_User]): Createdbyuser value
  - `CreatedByUserId` (Optional[str]): Createdbyuserid value
  - `StudyId` (Optional[str]): Studyid value
  - `ChangeSetId` (Optional[str]): Changesetid value
  - `Models` (Optional[list[str]]): List of Models
  - `SimulationOptions` (Optional[Simulation_SimulationOption]): Simulationoptions value
  - `SimulationAffinity` (Optional[Simulation_SimulationAffinity]): Simulationaffinity value
  - `ParallelizationOptions` (Optional[Simulation_ParallelizationOption]): Parallelizationoptions value
  - `SimulationData` (Optional[list[Simulation_SimulationDataUri]]): List of Simulationdata
  - `SolutionData` (Optional[list[Simulation_SolutionData]]): List of Solutiondata
  - `DataConfiguration` (Optional[Simulation_DataConfiguration]): Dataconfiguration value
  - `SimulationEngine` (Optional[Simulation_SimulationEngine]): Simulationengine value
  - `Tags` (Optional[dict[str, str]]): Tags value
  - `Source` (Optional[str]): Source value
  - `Priority` (Optional[int]): Priority value
  - `RequestedCpuCores` (Optional[int]): Requestedcpucores value
  - `MinimumMemoryInGb` (Optional[float]): Minimummemoryingb value
  - `QueueId` (Optional[str]): Queueid value
  - `RunParameters` (Optional[list[Simulation_RunParameters]]): List of Runparameters
  - `RequestedInstanceType` (Optional[Simulation_WorkerPoolInstance]): Requestedinstancetype value
  - `SimulationType` (Optional[Simulation_SimulationTypeEnum]): Simulationtype value

- `SimulationStarted` (Optional[list[Contracts_SimulationStarted]]): List of Simulationstarted (list)

- ▶ Properties of `Contracts_SimulationStarted`
  - `Id` (Optional[GuidValue]): Id value
  - `ExecutionId` (Optional[GuidValue]): Executionid value
  - `CreatedDate` (Optional[str]): Createddate value
  - `Status` (Optional[str]): Status value

```
file_path = r"c:\path\to\file.txt"

simulation_enqueue_simulation_resp:
list[CommandResponse[Contracts_EnqueueSimulationResponse]] =
pxc.simulation.enqueue_simulation(file_path=file_path, print_message=True)
simulation_enqueue_simulation_final: Contracts_EnqueueSimulationResponse =
SDKBase.get_response_data(simulation_enqueue_simulation_resp)

if simulation_enqueue_simulation_final is not None:
    # Iterate over SimulationStarted list
    if simulation_enqueue_simulation_final.SimulationStarted is not None:
        for item in simulation_enqueue_simulation_final.SimulationStarted:
            # Access properties of Contracts_SimulationStarted object
            print(f"Id: {item.Id}")
            print(f"ExecutionId: {item.ExecutionId}")
            print(f"CreatedDate: {item.CreatedDate}")
            print(f"Status: {item.Status}")
    else:
        print(f"No SimulationStarted returned")
    # Access single properties
    if simulation_enqueue_simulation_final.EnqueueSimulationRequest is not None:
        print(f"EnqueueSimulationRequest:
{simulation_enqueue_simulation_final.EnqueueSimulationRequest}")
else:
    print(f"enqueue_simulation failed:
{simulation_enqueue_simulation_resp.Message}")
```

List Simulation Engines

Description: Execute list_simulation_engines operation

Parameters:

- `optimization_engine_type` (str): optimization_engine_type parameter (optional)

Response Structure:

- `SimulationEngines` (Optional[list[Contracts_AvailableEngine]]): List of Simulationengines (list)

- ▶ Properties of `Contracts_AvailableEngine`
  - `Id` (Optional[str]): Id value
  - `Version` (Optional[str]): Version value
  - `DisplayName` (Optional[str]): Displayname value
  - `Description` (Optional[str]): Description value

    - `Status` (Optional[str]): Status value
    - `ReleasedDate` (Optional[str]): Releaseddate value
    - `OperatingSystem` (Optional[str]): Operatingsystem value
    - `EngineType` (Optional[str]): Enginetype value
    - `OptimizationEngine` (Optional[str]): Optimizationengine value
    - `IsBeta` (Optional[bool]): Isbeta value
    - `IsAvailableToAll` (Optional[bool]): Isavailabletoall value

```
optimization_engine_type = "PLEXOS"

simulation_list_simulation_engines_resp:
list[CommandResponse[Contracts_ListSimulationEngineResponse]] =
pxc.simulation.list_simulation_engines(optimization_engine_type=optimization_engin
e_type, print_message=True)
simulation_list_simulation_engines_final: Contracts_ListSimulationEngineResponse =
SDKBase.get_response_data(simulation_list_simulation_engines_resp)

if simulation_list_simulation_engines_final is not None:
    # Iterate over SimulationEngines list
    if simulation_list_simulation_engines_final.SimulationEngines is not None:
        for item in simulation_list_simulation_engines_final.SimulationEngines:
            # Access properties of Contracts_AvailableEngine object
            print(f"Id: {item.Id}")
            print(f"Version: {item.Version}")
            print(f"DisplayName: {item.DisplayName}")
            print(f"Description: {item.Description}")
            print(f"Status: {item.Status}")
            # ... and 6 more properties
    else:
        print(f"No SimulationEngines returned")
else:
    print(f"list_simulation_engines failed:
{simulation_list_simulation_engines_resp.Message}")
```

List Simulation Pool Capability

Description: Execute list_simulation_pool_capability operation

Response Structure:

- `SimulationPoolCapabilities` (Optional[list[Contracts_SimulationPoolCapability]]): List of Simulationpoolcapabilities (list)

  ▶ Properties of `Contracts_SimulationPoolCapability`
    - `Type` (Optional[str]): Type value
    - `Cores` (Optional[int]): Cores value
    - `Memory` (Optional[float]): Memory value
    - `BaseClockSpeed` (Optional[float]): Baseclockspeed value
    - `OperatingSystem` (Optional[str]): Operatingsystem value

    - `Capacity` (Optional[int]): Capacity value

```
simulation_list_simulation_pool_capability_resp:
list[CommandResponse[Contracts_ListSimulationPoolCapabilityResponse]] =
pxc.simulation.list_simulation_pool_capability(print_message=True)
simulation_list_simulation_pool_capability_final:
Contracts_ListSimulationPoolCapabilityResponse =
SDKBase.get_response_data(simulation_list_simulation_pool_capability_resp)

if simulation_list_simulation_pool_capability_final is not None:
    # Iterate over SimulationPoolCapabilities list
    if simulation_list_simulation_pool_capability_final.SimulationPoolCapabilities
is not None:
        for item in
simulation_list_simulation_pool_capability_final.SimulationPoolCapabilities:
            # Access properties of Contracts_SimulationPoolCapability object
            print(f"Type: {item.Type}")
            print(f"Cores: {item.Cores}")
            print(f"Memory: {item.Memory}")
            print(f"BaseClockSpeed: {item.BaseClockSpeed}")
            print(f"OperatingSystem: {item.OperatingSystem}")
            # ... and 1 more properties
    else:
        print(f"No SimulationPoolCapabilities returned")
else:
    print(f"list_simulation_pool_capability failed:
{simulation_list_simulation_pool_capability_resp.Message}")
```

Solution

Archive Solution

Description: Execute archive_solution operation

Parameters:

- `execution_id` (uuid4()): Unique identifier for a specific execution

Response Structure:

- `SolutionId` (Optional[str]): Solutionid value (single)
- `SolutionStatus` (Optional[str]): Solutionstatus value (single)
- `ExecutionId` (Optional[str]): Executionid value (single)

```
execution_id = "550e8400-e29b-41d4-a716-446655440000"

solution_archive_solution_resp:
list[CommandResponse[Contracts_SolutionStatusCommandResponse]] =
pxc.solution.archive_solution(execution_id=execution_id, print_message=True)
```

```
solution_archive_solution_final: Contracts_SolutionStatusCommandResponse =
SDKBase.get_response_data(solution_archive_solution_resp)

if solution_archive_solution_final is not None:
    # Access single properties
    if solution_archive_solution_final.SolutionId is not None:
        print(f"SolutionId: {solution_archive_solution_final.SolutionId}")
    if solution_archive_solution_final.SolutionStatus is not None:
        print(f"SolutionStatus: {solution_archive_solution_final.SolutionStatus}")
    if solution_archive_solution_final.ExecutionId is not None:
        print(f"ExecutionId: {solution_archive_solution_final.ExecutionId}")
else:
    print(f"archive_solution failed: {solution_archive_solution_resp.Message}")
```

Delete Solution

Description: Execute delete_solution operation

Parameters:

- `execution_id` (uuid4()): Unique identifier for a specific execution

Response Structure:

- `SolutionId` (Optional[str]): Solutionid value (single)
- `SolutionStatus` (Optional[str]): Solutionstatus value (single)
- `ExecutionId` (Optional[str]): Executionid value (single)

```
execution_id = "550e8400-e29b-41d4-a716-446655440000"

solution_delete_solution_resp:
list[CommandResponse[Contracts_SolutionStatusCommandResponse]] =
pxc.solution.delete_solution(execution_id=execution_id, print_message=True)
solution_delete_solution_final: Contracts_SolutionStatusCommandResponse =
SDKBase.get_response_data(solution_delete_solution_resp)

if solution_delete_solution_final is not None:
    # Access single properties
    if solution_delete_solution_final.SolutionId is not None:
        print(f"SolutionId: {solution_delete_solution_final.SolutionId}")
    if solution_delete_solution_final.SolutionStatus is not None:
        print(f"SolutionStatus: {solution_delete_solution_final.SolutionStatus}")
    if solution_delete_solution_final.ExecutionId is not None:
        print(f"ExecutionId: {solution_delete_solution_final.ExecutionId}")
else:
    print(f"delete_solution failed: {solution_delete_solution_resp.Message}")
```

Get Solution Id

Description: Execute get_solution_id operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study
- `model_name` (str): model_name parameter

Response Structure:

- `ChangesetId` (Optional[str]): Changesetid value (single)
- `SolutionId` (Optional[str]): Solutionid value (single)

```
study_id = "550e8400-e29b-41d4-a716-446655440000"
model_name = "MyModel"

solution_get_solution_id_resp:
list[CommandResponse[Contracts_GetSolutionIdResponse]] =
pxc.solution.get_solution_id(study_id=study_id, model_name=model_name,
print_message=True)
solution_get_solution_id_final: Contracts_GetSolutionIdResponse =
SDKBase.get_response_data(solution_get_solution_id_resp)

if solution_get_solution_id_final is not None:
    # Access single properties
    if solution_get_solution_id_final.ChangesetId is not None:
        print(f"ChangesetId: {solution_get_solution_id_final.ChangesetId}")
    if solution_get_solution_id_final.SolutionId is not None:
        print(f"SolutionId: {solution_get_solution_id_final.SolutionId}")
else:
    print(f"get_solution_id failed: {solution_get_solution_id_resp.Message}")
```

List Solution Reports

Description: Execute list_solution_reports operation

Response Structure:

- `SolutionReports` (Optional[list[Contracts_SolutionReport]]): List of Solutionreports (list)

  ▶ Properties of `Contracts_SolutionReport`
    - `ReportId` (Optional[str]): Reportid value
    - `ReportName` (Optional[str]): Reportname value

```


solution_list_solution_reports_resp:
list[CommandResponse[Contracts_ListSolutionReportResponse]] =
pxc.solution.list_solution_reports(print_message=True)
solution_list_solution_reports_final: Contracts_ListSolutionReportResponse =
SDKBase.get_response_data(solution_list_solution_reports_resp)

if solution_list_solution_reports_final is not None:
    # Iterate over SolutionReports list
    if solution_list_solution_reports_final.SolutionReports is not None:
        for item in solution_list_solution_reports_final.SolutionReports:
            # Access properties of Contracts_SolutionReport object
            print(f"ReportId: {item.ReportId}")
            print(f"ReportName: {item.ReportName}")
    else:
        print(f"No SolutionReports returned")
else:
    print(f"list_solution_reports failed:
{solution_list_solution_reports_resp.Message}")
```

List Solutions

Description: Execute list_solutions operation

Parameters:

- `solution_id` (uuid4()): solution_id parameter (optional)
- `study_id` (uuid4()): Unique identifier for a specific study (optional, default: None)
- `simulation_id` (uuid4()): Unique identifier for a specific simulation (optional, default: None)
- `execution_id` (uuid4()): Unique identifier for a specific execution (optional, default: None)
- `type` (str): type parameter (optional, default: None)
- `status` (str): status parameter (optional, default: None)
- `order_by` (str): Field to order results by (optional, default: None)
- `descending` (bool): Order results in descending order (optional, default: None)
- `top` (int): Maximum number of results to return (optional, default: None)
- `skip` (int): Number of results to skip (optional, default: None)
- `raw` (str): Raw filter string for advanced queries (optional, default: None)

Response Structure:

- `Solutions` (Optional[list[Contracts_Solution]]): List of Solutions (list)

  ▶ Properties of `Contracts_Solution`
    - `SolutionId` (Optional[GuidValue]): Solutionid value
    - `StudyId` (Optional[GuidValue]): Studyid value
    - `ExecutionId` (Optional[GuidValue]): Executionid value
    - `SimulationId` (Optional[GuidValue]): Simulationid value
    - `Type` (Optional[str]): Type value
    - `Status` (Optional[str]): Status value
    - `LastUpdatedDate` (Optional[str]): Lastupdateddate value
    - `CreatedDate` (Optional[str]): Createddate value
    - `ModelName` (Optional[str]): Modelname value
    - `TypeVersion` (Optional[int]): Typeversion value

```
solution_id = None
study_id = None
simulation_id = None
execution_id = None
```

```
type = "Standard"
status = "Active"
order_by = "CreatedAt"
descending = True
top = 10
skip = 0
raw = "filter expression"

solution_list_solutions_resp:
list[CommandResponse[Contracts_ListSolutionsResponse]] =
pxc.solution.list_solutions(solution_id=solution_id, study_id=study_id,
simulation_id=simulation_id, execution_id=execution_id, type=type, status=status,
order_by=order_by, descending=descending, top=top, skip=skip, raw=raw,
print_message=True)
solution_list_solutions_final: Contracts_ListSolutionsResponse =
SDKBase.get_response_data(solution_list_solutions_resp)

if solution_list_solutions_final is not None:
    # Iterate over Solutions list
    if solution_list_solutions_final.Solutions is not None:
        for item in solution_list_solutions_final.Solutions:
            # Access properties of Contracts_Solution object
            print(f"SolutionId: {item.SolutionId}")
            print(f"StudyId: {item.StudyId}")
            print(f"ExecutionId: {item.ExecutionId}")
            print(f"SimulationId: {item.SimulationId}")
            print(f"Type: {item.Type}")
            # ... and 5 more properties
    else:
        print(f"No Solutions returned")
else:
    print(f"list_solutions failed: {solution_list_solutions_resp.Message}")
```

Solution Reports

Description: Execute solution_reports operation

Parameters:

- `solution_id` (uuid4()): solution_id parameter
- `report_id` (str): report_id parameter
- `output_directory` (str): Local directory to save downloaded files
- `file` (str): file parameter (optional)

Response Structure: None

```
solution_id = "550e8400-e29b-41d4-a716-446655440000"
report_id = "550e8400-e29b-41d4-a716-446655440000"
output_directory = r"c:\output"
file = "data.csv"
```

```
solution_solution_reports_resp:
list[CommandResponse[Contracts_SolutionReportsEmptyResponse]] =
pxc.solution.solution_reports(solution_id=solution_id, report_id=report_id,
output_directory=output_directory, file=file, print_message=True)
solution_solution_reports_final: Contracts_SolutionReportsEmptyResponse =
SDKBase.get_response_data(solution_solution_reports_resp)

if solution_solution_reports_final is not None:
    print(f"solution_reports completed successfully")
else:
    print(f"solution_reports failed: {solution_solution_reports_resp.Message}")
```

Solution Stitching

Description: Execute solution_stitching operation

Parameters:

- `execution_id` (uuid4()): Unique identifier for a specific execution
- `number_of_cores` (int): number_of_cores parameter
- `memory_in_gb` (float): memory_in_gb parameter

Response Structure:

- `Data` (Optional[Simulation_EnqueuedSimulation]): Data value (single)

  ▶ Properties of `Simulation_EnqueuedSimulation`
    - `Id` (Optional[str]): Id value
    - `CreatedAt` (Optional[str]): Createdat value
    - `Status` (Optional[str]): Status value
    - `ExecutionId` (Optional[str]): Executionid value

```
execution_id = "550e8400-e29b-41d4-a716-446655440000"
number_of_cores = 4
memory_in_gb = 8.0

solution_solution_stitching_resp:
list[CommandResponse[Contracts_SolutionStitchingResponse]] =
pxc.solution.solution_stitching(execution_id=execution_id,
number_of_cores=number_of_cores, memory_in_gb=memory_in_gb, print_message=True)
solution_solution_stitching_final: Contracts_SolutionStitchingResponse =
SDKBase.get_response_data(solution_solution_stitching_resp)

if solution_solution_stitching_final is not None:
    # Access single properties
    if solution_solution_stitching_final.Data is not None:
        print(f"Data: {solution_solution_stitching_final.Data}")
else:
    print(f"solution_stitching failed:
{solution_solution_stitching_resp.Message}")
```

Unarchive Solution

Description: Execute unarchive_solution operation

Parameters:

- `execution_id` (uuid4()): Unique identifier for a specific execution

Response Structure:

- `SolutionId` (Optional[str]): Solutionid value (single)
- `SolutionStatus` (Optional[str]): Solutionstatus value (single)
- `ExecutionId` (Optional[str]): Executionid value (single)

```
execution_id = "550e8400-e29b-41d4-a716-446655440000"

solution_unarchive_solution_resp:
list[CommandResponse[Contracts_SolutionStatusCommandResponse]] =
pxc.solution.unarchive_solution(execution_id=execution_id, print_message=True)
solution_unarchive_solution_final: Contracts_SolutionStatusCommandResponse =
SDKBase.get_response_data(solution_unarchive_solution_resp)

if solution_unarchive_solution_final is not None:
    # Access single properties
    if solution_unarchive_solution_final.SolutionId is not None:
        print(f"SolutionId: {solution_unarchive_solution_final.SolutionId}")
    if solution_unarchive_solution_final.SolutionStatus is not None:
        print(f"SolutionStatus:
{solution_unarchive_solution_final.SolutionStatus}")
    if solution_unarchive_solution_final.ExecutionId is not None:
        print(f"ExecutionId: {solution_unarchive_solution_final.ExecutionId}")
else:
    print(f"unarchive_solution failed:
{solution_unarchive_solution_resp.Message}")
```

Get Solution Data Using View

Description: Execute get_solution_data_using_view operation

Parameters:

- `output_directory` (str): Local directory to save downloaded files
- `solution_id` (uuid4()): solution_id parameter
- `report_id` (str): report_id parameter
- `view_id` (str): view_id parameter
- `overwrite` (bool): overwrite parameter (optional)
- `file` (str): file parameter (optional)

Response Structure:

- `FilePath` (Optional[str]): Filepath value (single)

```
output_directory = r"c:\output"
solution_id = "550e8400-e29b-41d4-a716-446655440000"
report_id = "550e8400-e29b-41d4-a716-446655440000"
view_id = "550e8400-e29b-41d4-a716-446655440000"
overwrite = True
file = "data.csv"

solution_get_solution_data_using_view_resp:
list[CommandResponse[Contracts_GetSolutionReportDataResponse]] =
pxc.solution.get_solution_data_using_view(output_directory=output_directory,
solution_id=solution_id, report_id=report_id, view_id=view_id,
overwrite=overwrite, file=file, print_message=True)
solution_get_solution_data_using_view_final:
Contracts_GetSolutionReportDataResponse =
SDKBase.get_response_data(solution_get_solution_data_using_view_resp)

if solution_get_solution_data_using_view_final is not None:
    # Access single properties
    if solution_get_solution_data_using_view_final.FilePath is not None:
        print(f"FilePath: {solution_get_solution_data_using_view_final.FilePath}")
else:
    print(f"get_solution_data_using_view failed:
{solution_get_solution_data_using_view_resp.Message}")
```

Get View Reports Details

Description: Execute get_view_reports_details operation

Parameters:

- `view_id` (str): view_id parameter

Response Structure:

- `ViewReportDetials` (Optional[Solution_ViewReportDetails]): Viewreportdetials value (single)

```
view_id = "550e8400-e29b-41d4-a716-446655440000"

solution_get_view_reports_details_resp:
list[CommandResponse[Contracts_GetViewReportsDetailsResponse]] =
pxc.solution.get_view_reports_details(view_id=view_id, print_message=True)
solution_get_view_reports_details_final: Contracts_GetViewReportsDetailsResponse =
SDKBase.get_response_data(solution_get_view_reports_details_resp)

if solution_get_view_reports_details_final is not None:
    # Access single properties
    if solution_get_view_reports_details_final.ViewReportDetials is not None:
        print(f"ViewReportDetials:
{solution_get_view_reports_details_final.ViewReportDetials}")
else:
    print(f"get_view_reports_details failed:
{solution_get_view_reports_details_resp.Message}")
```

Publish View

Description: Execute publish_view operation

Parameters:

- `view_file_path` (str): view_file_path parameter

Response Structure:

- `ViewId` (Optional[str]): Viewid value (single)

```
view_file_path = r"c:\path\to\view.json"

solution_publish_view_resp: list[CommandResponse[Contracts_PublishViewResponse]] =
pxc.solution.publish_view(view_file_path=view_file_path, print_message=True)
solution_publish_view_final: Contracts_PublishViewResponse =
SDKBase.get_response_data(solution_publish_view_resp)

if solution_publish_view_final is not None:
    # Access single properties
    if solution_publish_view_final.ViewId is not None:
        print(f"ViewId: {solution_publish_view_final.ViewId}")
else:
    print(f"publish_view failed: {solution_publish_view_resp.Message}")
```

Download Solution

Description: Execute download_solution operation

Parameters:

- `solution_id` (uuid4()): solution_id parameter
- `output_directory` (str): Local directory to save downloaded files
- `solution_type` (str): solution_type parameter (optional)
- `overwrite` (bool): overwrite parameter (optional)
- `file_name` (str): file_name parameter (optional)
- `generate_metadata` (bool): generate_metadata parameter (optional, default: None)
- `metadata_file_name` (str): metadata_file_name parameter (optional, default: None)

Response Structure:

- `files` (Optional[list[str]]): List of Files (list)
- `SolutionId` (Optional[str]): Solutionid value (single)
- `IsDownloadSuccessful` (Optional[bool]): Isdownloadsuccessful value (single)

```
solution_id = "550e8400-e29b-41d4-a716-446655440000"
output_directory = r"c:\output"
solution_type = "Standard"
overwrite = True
file_name = "output.txt"
generate_metadata = True
metadata_file_name = "metadata.json"

solution_download_solution_resp: list[CommandResponse[Contracts_DownloadSolution]]
= pxc.solution.download_solution(solution_id=solution_id,
output_directory=output_directory, solution_type=solution_type,
overwrite=overwrite, file_name=file_name, generate_metadata=generate_metadata,
metadata_file_name=metadata_file_name, print_message=True)
solution_download_solution_final: Contracts_DownloadSolution =
SDKBase.get_response_data(solution_download_solution_resp)

if solution_download_solution_final is not None:
    # Iterate over files list
    if solution_download_solution_final.files is not None:
        for item in solution_download_solution_final.files:
            print(item)
    else:
        print(f"No files returned")
    # Access single properties
    if solution_download_solution_final.SolutionId is not None:
        print(f"SolutionId: {solution_download_solution_final.SolutionId}")
    if solution_download_solution_final.IsDownloadSuccessful is not None:
        print(f"IsDownloadSuccessful:
{solution_download_solution_final.IsDownloadSuccessful}")
else:
    print(f"download_solution failed: {solution_download_solution_resp.Message}")
```

List Solution Files

Description: Execute list_solution_files operation

Parameters:

- `solution_id` (uuid4()): solution_id parameter
- `solution_type` (str): solution_type parameter (optional)
- `include_archive_entries` (bool): include_archive_entries parameter (optional)

Response Structure:

- `ConsoleSolutionTypeFileLists` (Optional[list[Solution_ConsoleSolutionTypeFileList]]): List of Consolesolutiontypefilelists (list)

```
solution_id = "550e8400-e29b-41d4-a716-446655440000"
solution_type = "Standard"
include_archive_entries = True
```

```
solution_list_solution_files_resp:
list[CommandResponse[Contracts_ListSolutionFile]] =
pxc.solution.list_solution_files(solution_id=solution_id,
solution_type=solution_type, include_archive_entries=include_archive_entries,
print_message=True)
solution_list_solution_files_final: Contracts_ListSolutionFile =
SDKBase.get_response_data(solution_list_solution_files_resp)

if solution_list_solution_files_final is not None:
    # Iterate over ConsoleSolutionTypeFileLists list
    if solution_list_solution_files_final.ConsoleSolutionTypeFileLists is not
None:
        for item in
solution_list_solution_files_final.ConsoleSolutionTypeFileLists:
            print(item)
    else:
        print(f"No ConsoleSolutionTypeFileLists returned")
else:
    print(f"list_solution_files failed:
{solution_list_solution_files_resp.Message}")
```

List Solution File Types

Description: Execute list_solution_file_types operation

Parameters:

- `solution_id` (uuid4()): solution_id parameter

Response Structure:

- `FileTypes` (Optional[list[str]]): List of Filetypes (list)

```
solution_id = "550e8400-e29b-41d4-a716-446655440000"

solution_list_solution_file_types_resp:
list[CommandResponse[Contracts_ListSolutionFileTypes]] =
pxc.solution.list_solution_file_types(solution_id=solution_id, print_message=True)
solution_list_solution_file_types_final: Contracts_ListSolutionFileTypes =
SDKBase.get_response_data(solution_list_solution_file_types_resp)

if solution_list_solution_file_types_final is not None:
    # Iterate over FileTypes list
    if solution_list_solution_file_types_final.FileTypes is not None:
        for item in solution_list_solution_file_types_final.FileTypes:
            print(item)
    else:
        print(f"No FileTypes returned")
else:
    print(f"list_solution_file_types failed:
{solution_list_solution_file_types_resp.Message}")
```

Convert Hybrid To Parquet

Description: Execute convert_hybrid_to_parquet operation

Parameters:

- `sql_lite_path` (str): sql_lite_path parameter
- `parquet_directory` (str): parquet_directory parameter
- `output_directory` (str): Local directory to save downloaded files

Response Structure:

- `Response` (Optional[int]): Response value (single)

```
sql_lite_path = r"c:\path\to\data.db"
parquet_directory = r"c:\path\to\parquet"
output_directory = r"c:\output"

solution_convert_hybrid_to_parquet_resp:
list[CommandResponse[Contracts_ConvertHybridToParquetResponse]] =
pxc.solution.convert_hybrid_to_parquet(sql_lite_path=sql_lite_path,
parquet_directory=parquet_directory, output_directory=output_directory,
print_message=True)
solution_convert_hybrid_to_parquet_final: Contracts_ConvertHybridToParquetResponse
= SDKBase.get_response_data(solution_convert_hybrid_to_parquet_resp)

if solution_convert_hybrid_to_parquet_final is not None:
    # Access single properties
    if solution_convert_hybrid_to_parquet_final.Response is not None:
        print(f"Response: {solution_convert_hybrid_to_parquet_final.Response}")
else:
    print(f"convert_hybrid_to_parquet failed:
{solution_convert_hybrid_to_parquet_resp.Message}")
```

Convert Raw Zip To Hybrid

Description: Execute convert_raw_zip_to_hybrid operation

Parameters:

- `zip_path` (str): zip_path parameter
- `output_directory` (str): Local directory to save downloaded files
- `schema_version` (int): schema_version parameter (optional)

Response Structure:

- `Response` (Optional[int]): Response value (single)

```
zip_path = r"c:\path\to\data.zip"
output_directory = r"c:\output"
schema_version = 1

solution_convert_raw_zip_to_hybrid_resp:
list[CommandResponse[Contracts_ConvertRawZipToHybridResponse]] =
pxc.solution.convert_raw_zip_to_hybrid(zip_path=zip_path,
output_directory=output_directory, schema_version=schema_version,
print_message=True)
solution_convert_raw_zip_to_hybrid_final: Contracts_ConvertRawZipToHybridResponse
= SDKBase.get_response_data(solution_convert_raw_zip_to_hybrid_resp)

if solution_convert_raw_zip_to_hybrid_final is not None:
    # Access single properties
    if solution_convert_raw_zip_to_hybrid_final.Response is not None:
        print(f"Response: {solution_convert_raw_zip_to_hybrid_final.Response}")
else:
    print(f"convert_raw_zip_to_hybrid failed:
{solution_convert_raw_zip_to_hybrid_resp.Message}")
```

Convert Raw Zip To Parquet

Description: Execute convert_raw_zip_to_parquet operation

Parameters:

- `zip_path` (str): zip_path parameter
- `output_directory` (str): Local directory to save downloaded files
- `parquet_schema_version` (int): parquet_schema_version parameter (optional)

Response Structure:

- `Response` (Optional[int]): Response value (single)

```
zip_path = r"c:\path\to\data.zip"
output_directory = r"c:\output"
parquet_schema_version = 1

solution_convert_raw_zip_to_parquet_resp:
list[CommandResponse[Contracts_ConvertRawZipToParquetResponse]] =
pxc.solution.convert_raw_zip_to_parquet(zip_path=zip_path,
output_directory=output_directory, parquet_schema_version=parquet_schema_version,
print_message=True)
solution_convert_raw_zip_to_parquet_final:
Contracts_ConvertRawZipToParquetResponse =
SDKBase.get_response_data(solution_convert_raw_zip_to_parquet_resp)

if solution_convert_raw_zip_to_parquet_final is not None:
    # Access single properties
    if solution_convert_raw_zip_to_parquet_final.Response is not None:
        print(f"Response: {solution_convert_raw_zip_to_parquet_final.Response}")
```

```
else:
    print(f"convert_raw_zip_to_parquet failed: 
{solution_convert_raw_zip_to_parquet_resp.Message}")
```

Study

Archive Study

Description: Execute archive_study operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study
- `delete_solutions` (bool): delete_solutions parameter (optional)

Response Structure:

- `StudyId` (Optional[str]): Studyid value (single)

```
study_id = "550e8400-e29b-41d4-a716-446655440000"
delete_solutions = True

study_archive_study_resp: list[CommandResponse[Contracts_StudyCommandResponse]] = 
pxc.study.archive_study(study_id=study_id, delete_solutions=delete_solutions, 
print_message=True)
study_archive_study_final: Contracts_StudyCommandResponse = 
SDKBase.get_response_data(study_archive_study_resp)

if study_archive_study_final is not None:
    # Access single properties
    if study_archive_study_final.StudyId is not None:
        print(f"StudyId: {study_archive_study_final.StudyId}")
else:
    print(f"archive_study failed: {study_archive_study_resp.Message}")
```

Clone Study

Description: Clone an existing study

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study
- `output_directory_path` (str): output_directory_path parameter

Response Structure:

- `StudyId` (Optional[str]): Studyid value (single)
- `OutputPath` (Optional[str]): Outputpath value (single)
- `StudyName` (Optional[str]): Studyname value (single)

```
study_id = "550e8400-e29b-41d4-a716-446655440000"
output_directory_path = r"c:\output"

study_clone_study_resp: list[CommandResponse[Contracts_CloneStudyResponse]] =
pxc.study.clone_study(study_id=study_id,
output_directory_path=output_directory_path, print_message=True)
study_clone_study_final: Contracts_CloneStudyResponse =
SDKBase.get_response_data(study_clone_study_resp)

if study_clone_study_final is not None:
    # Access single properties
    if study_clone_study_final.StudyId is not None:
        print(f"StudyId: {study_clone_study_final.StudyId}")
    if study_clone_study_final.OutputPath is not None:
        print(f"OutputPath: {study_clone_study_final.OutputPath}")
    if study_clone_study_final.StudyName is not None:
        print(f"StudyName: {study_clone_study_final.StudyName}")
else:
    print(f"clone_study failed: {study_clone_study_resp.Message}")
```

Create Study

Description: Create a new study

Parameters:

- `study_name` (str): study_name parameter
- `study_description` (str): study_description parameter
- `study_db_path` (str): study_db_path parameter

Response Structure:

- `StudyId` (Optional[str]): Studyid value (single)

```
study_name = "My Study"
study_description = "Study description"
study_db_path = r"c:\path\to\study.db"

study_create_study_resp: list[CommandResponse[Contracts_StudyCommandResponse]] =
pxc.study.create_study(study_name=study_name, study_description=study_description,
study_db_path=study_db_path, print_message=True)
study_create_study_final: Contracts_StudyCommandResponse =
SDKBase.get_response_data(study_create_study_resp)

if study_create_study_final is not None:
    # Access single properties
    if study_create_study_final.StudyId is not None:
        print(f"StudyId: {study_create_study_final.StudyId}")
else:
    print(f"create_study failed: {study_create_study_resp.Message}")
```

Delete Local Study

Description: Execute delete_local_study operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study
- `full_delete` (bool): full_delete parameter (optional)

Response Structure:

- `StudyId` (Optional[str]): Studyid value (single)

```
study_id = "550e8400-e29b-41d4-a716-446655440000"
full_delete = True

study_delete_local_study_resp:
list[CommandResponse[Contracts_StudyCommandResponse]] =
pxc.study.delete_local_study(study_id=study_id, full_delete=full_delete,
print_message=True)
study_delete_local_study_final: Contracts_StudyCommandResponse =
SDKBase.get_response_data(study_delete_local_study_resp)

if study_delete_local_study_final is not None:
    # Access single properties
    if study_delete_local_study_final.StudyId is not None:
        print(f"StudyId: {study_delete_local_study_final.StudyId}")
else:
    print(f"delete_local_study failed: {study_delete_local_study_resp.Message}")
```

Delete Study

Description: Delete a study

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study

Response Structure:

- `StudyId` (Optional[str]): Studyid value (single)

```
study_id = "550e8400-e29b-41d4-a716-446655440000"

study_delete_study_resp: list[CommandResponse[Contracts_StudyCommandResponse]] =
pxc.study.delete_study(study_id=study_id, print_message=True)
study_delete_study_final: Contracts_StudyCommandResponse =
SDKBase.get_response_data(study_delete_study_resp)

if study_delete_study_final is not None:
    # Access single properties
    if study_delete_study_final.StudyId is not None:
        print(f"StudyId: {study_delete_study_final.StudyId}")
else:
    print(f"delete_study failed: {study_delete_study_resp.Message}")
```

Find Study

Description: Execute find_study operation

Parameters:

- `study_name` (str): study_name parameter

Response Structure:

- `Studies` (Optional[list[Contracts_Study]]): List of Studies (list)

  ▶ Properties of `Contracts_Study`
    - `Id` (Optional[GuidValue]): Id value
    - `Name` (Optional[str]): Name value
    - `Description` (Optional[str]): Description value
    - `Status` (Optional[str]): Status value
    - `LastUpdateMessage` (Optional[str]): Lastupdatemessage value
    - `CreatedDate` (Optional[str]): Createddate value
    - `LastUpdatedAtUtc` (Optional[str]): Lastupdatedatutc value
    - `StudyType` (Optional[str]): Studytype value
    - `isAccessibleToRequestingUser` (Optional[bool]): Isaccessibletorequestinguser value
    - `createdByUserId` (Optional[str]): Createdbyuserid value
    - `User` (Optional[Contracts_User]): User value

```
study_name = "My Study"

study_find_study_resp: list[CommandResponse[Contracts_ListStudiesResponse]] =
pxc.study.find_study(study_name=study_name, print_message=True)
study_find_study_final: Contracts_ListStudiesResponse =
SDKBase.get_response_data(study_find_study_resp)

if study_find_study_final is not None:
    # Iterate over Studies list
    if study_find_study_final.Studies is not None:
        for item in study_find_study_final.Studies:
            # Access properties of Contracts_Study object
            print(f"Id: {item.Id}")
            print(f"Name: {item.Name}")
            print(f"Description: {item.Description}")
            print(f"Status: {item.Status}")
            print(f"LastUpdateMessage: {item.LastUpdateMessage}")
            # ... and 6 more properties
    else:
        print(f"No Studies returned")
else:
    print(f"find_study failed: {study_find_study_resp.Message}")
```

Grant User Access

Description: Execute grant_user_access operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study
- `user_emails` (list[str]): user_emails parameter

Response Structure:

- `Users` (Optional[list[str]]): List of Users (list)

```
study_id = "550e8400-e29b-41d4-a716-446655440000"
user_emails = ["user@example.com", "admin@example.com"]

study_grant_user_access_resp:
list[CommandResponse[Contracts_GrantUserAccessResponse]] =
pxc.study.grant_user_access(study_id=study_id, user_emails=user_emails,
print_message=True)
study_grant_user_access_final: Contracts_GrantUserAccessResponse =
SDKBase.get_response_data(study_grant_user_access_resp)

if study_grant_user_access_final is not None:
    # Iterate over Users list
    if study_grant_user_access_final.Users is not None:
        for item in study_grant_user_access_final.Users:
            print(item)
    else:
        print(f"No Users returned")
else:
    print(f"grant_user_access failed: {study_grant_user_access_resp.Message}")
```

List Local Studies

Description: Execute list_local_studies operation

Response Structure:

- `StudyRecords` (Optional[list[Contracts_LocalStudyRecordResponse]]): List of Studyrecords (list)

  ▶ Properties of `Contracts_LocalStudyRecordResponse`
  - `StudyId` (Optional[str]): Studyid value
  - `StudyXmlPath` (Optional[str]): Studyxmlpath value

```
study_list_local_studies_resp:
list[CommandResponse[Contracts_ListLocalStudiesResponse]] =
pxc.study.list_local_studies(print_message=True)
study_list_local_studies_final: Contracts_ListLocalStudiesResponse =
SDKBase.get_response_data(study_list_local_studies_resp)

if study_list_local_studies_final is not None:
    # Iterate over StudyRecords list
    if study_list_local_studies_final.StudyRecords is not None:
        for item in study_list_local_studies_final.StudyRecords:
            # Access properties of Contracts_LocalStudyRecordResponse object
            print(f"StudyId: {item.StudyId}")
            print(f"StudyXmlPath: {item.StudyXmlPath}")
    else:
        print(f"No StudyRecords returned")
else:
    print(f"list_local_studies failed: {study_list_local_studies_resp.Message}")
```

List Studies

Description: List available studies

Parameters:

- `order_by` (str): Field to order results by (optional)
- `descending` (bool): Order results in descending order (optional, default: None)
- `top` (int): Maximum number of results to return (optional, default: None)
- `skip` (int): Number of results to skip (optional, default: None)
- `study_type` (str): study_type parameter (optional, default: None)
- `raw` (str): Raw filter string for advanced queries (optional, default: None)
- `filter_by_user_id` (bool): filter_by_user_id parameter (optional, default: None)

Response Structure:

- `Studies` (Optional[list[Contracts_Study]]): List of Studies (list)

  ▶ Properties of `Contracts_Study`
    - `Id` (Optional[GuidValue]): Id value
    - `Name` (Optional[str]): Name value
    - `Description` (Optional[str]): Description value
    - `Status` (Optional[str]): Status value
    - `LastUpdateMessage` (Optional[str]): Lastupdatemessage value
    - `CreatedDate` (Optional[str]): Createddate value
    - `LastUpdatedAtUtc` (Optional[str]): Lastupdatedatutc value
    - `StudyType` (Optional[str]): Studytype value
    - `isAccessibleToRequestingUser` (Optional[bool]): Isaccessibletorequestinguser value
    - `createdByUserId` (Optional[str]): Createdbyuserid value
    - `User` (Optional[Contracts_User]): User value

```
order_by = "CreatedAt"
descending = True
top = 10
skip = 0
study_type = "Standard"
raw = "filter expression"
filter_by_user_id = True

study_list_studies_resp: list[CommandResponse[Contracts_ListStudiesResponse]] =
pxc.study.list_studies(order_by=order_by, descending=descending, top=top,
skip=skip, study_type=study_type, raw=raw, filter_by_user_id=filter_by_user_id,
print_message=True)
study_list_studies_final: Contracts_ListStudiesResponse =
SDKBase.get_response_data(study_list_studies_resp)

if study_list_studies_final is not None:
    # Iterate over Studies list
    if study_list_studies_final.Studies is not None:
        for item in study_list_studies_final.Studies:
            # Access properties of Contracts_Study object
            print(f"Id: {item.Id}")
            print(f"Name: {item.Name}")
            print(f"Description: {item.Description}")
            print(f"Status: {item.Status}")
            print(f"LastUpdateMessage: {item.LastUpdateMessage}")
            # ... and 6 more properties
    else:
        print(f"No Studies returned")
else:
    print(f"list_studies failed: {study_list_studies_resp.Message}")
```

List Study Ids For Folder

Description: Execute list_study_ids_for_folder operation

Parameters:

- `study_directory_path` (str): study_directory_path parameter

Response Structure:

- `StudyIds` (Optional[list[str]]): List of Studyids (list)

```
study_directory_path = r"c:\path\to\study"

study_list_study_ids_for_folder_resp:
list[CommandResponse[Contracts_ListStudyIdsForFolderResponse]] =
pxc.study.list_study_ids_for_folder(study_directory_path=study_directory_path,
print_message=True)
study_list_study_ids_for_folder_final: Contracts_ListStudyIdsForFolderResponse =
SDKBase.get_response_data(study_list_study_ids_for_folder_resp)
```

```
if study_list_study_ids_for_folder_final is not None:
    # Iterate over StudyIds list
    if study_list_study_ids_for_folder_final.StudyIds is not None:
        for item in study_list_study_ids_for_folder_final.StudyIds:
            print(item)
    else:
        print(f"No StudyIds returned")
else:
    print(f"list_study_ids_for_folder failed:
{study_list_study_ids_for_folder_resp.Message}")
```

Reset Study

Description: Execute reset_study operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study

Response Structure:

- `StudyId` (Optional[str]): Studyid value (single)
- `CloudStudyName` (Optional[str]): Cloudstudyname value (single)
- `Success` (Optional[bool]): Success value (single)
- `OutputPath` (Optional[str]): Outputpath value (single)

```
study_id = "550e8400-e29b-41d4-a716-446655440000"

study_reset_study_resp: list[CommandResponse[Contracts_ResetStudyResponse]] =
pxc.study.reset_study(study_id=study_id, print_message=True)
study_reset_study_final: Contracts_ResetStudyResponse =
SDKBase.get_response_data(study_reset_study_resp)

if study_reset_study_final is not None:
    # Access single properties
    if study_reset_study_final.StudyId is not None:
        print(f"StudyId: {study_reset_study_final.StudyId}")
    if study_reset_study_final.CloudStudyName is not None:
        print(f"CloudStudyName: {study_reset_study_final.CloudStudyName}")
    if study_reset_study_final.Success is not None:
        print(f"Success: {study_reset_study_final.Success}")
    if study_reset_study_final.OutputPath is not None:
        print(f"OutputPath: {study_reset_study_final.OutputPath}")
else:
    print(f"reset_study failed: {study_reset_study_resp.Message}")
```

Study Repair

Description: Execute study_repair operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study
- `database_file_path` (str): database_file_path parameter (optional)

Response Structure: None

```
study_id = "550e8400-e29b-41d4-a716-446655440000"
database_file_path = r"c:\path\to\database.db"

study_study_repair_resp: list[CommandResponse[Contracts_RepairStudyResponse]] =
pxc.study.study_repair(study_id=study_id, database_file_path=database_file_path,
print_message=True)
study_study_repair_final: Contracts_RepairStudyResponse =
SDKBase.get_response_data(study_study_repair_resp)

if study_study_repair_final is not None:
    print(f"study_repair completed successfully")
else:
    print(f"study_repair failed: {study_study_repair_resp.Message}")
```

Unarchive Study

Description: Execute unarchive_study operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study

Response Structure:

- `StudyId` (Optional[str]): Studyid value (single)

```
study_id = "550e8400-e29b-41d4-a716-446655440000"

study_unarchive_study_resp: list[CommandResponse[Contracts_StudyCommandResponse]]
= pxc.study.unarchive_study(study_id=study_id, print_message=True)
study_unarchive_study_final: Contracts_StudyCommandResponse =
SDKBase.get_response_data(study_unarchive_study_resp)

if study_unarchive_study_final is not None:
    # Access single properties
    if study_unarchive_study_final.StudyId is not None:
        print(f"StudyId: {study_unarchive_study_final.StudyId}")
else:
    print(f"unarchive_study failed: {study_unarchive_study_resp.Message}")
```

Validate Study Data

Description: Execute validate_study_data operation

Parameters:

- `database_file_path` (str): database_file_path parameter

Response Structure: None

```
database_file_path = r"c:\path\to\database.db"

study_validate_study_data_resp:
list[CommandResponse[Contracts_ValidateStudyResponse]] =
pxc.study.validate_study_data(database_file_path=database_file_path,
print_message=True)
study_validate_study_data_final: Contracts_ValidateStudyResponse =
SDKBase.get_response_data(study_validate_study_data_resp)

if study_validate_study_data_final is not None:
    print(f"validate_study_data completed successfully")
else:
    print(f"validate_study_data failed: {study_validate_study_data_resp.Message}")
```

Get Geocoded Objects

Description: Execute get_geocoded_objects operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study
- `changeset_id` (uuid4()): Unique identifier for a specific changeset

Response Structure:

- `Metric` (Optional[StudyStats_Metric]): Metric value (single)

```
study_id = "550e8400-e29b-41d4-a716-446655440000"
changeset_id = "550e8400-e29b-41d4-a716-446655440000"

study_get_geocoded_objects_resp:
list[CommandResponse[Contracts_GeocodedObjectsResponse]] =
pxc.study.get_geocoded_objects(study_id=study_id, changeset_id=changeset_id,
print_message=True)
study_get_geocoded_objects_final: Contracts_GeocodedObjectsResponse =
SDKBase.get_response_data(study_get_geocoded_objects_resp)

if study_get_geocoded_objects_final is not None:
    # Access single properties
    if study_get_geocoded_objects_final.Metric is not None:
        print(f"Metric: {study_get_geocoded_objects_final.Metric}")
else:
    print(f"get_geocoded_objects failed:
{study_get_geocoded_objects_resp.Message}")
```

Add Configurations

Description: Execute add_configurations operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study
- `study_setting_id` (str): study_setting_id parameter
- `study_setting_type` (str): study_setting_type parameter
- `settings_file_path` (str): settings_file_path parameter
- `show_settings_example` (bool): show_settings_example parameter (optional)

Response Structure:

- `StudyIds` (Optional[list[str]]): List of Studyids (list)

```
study_id = "550e8400-e29b-41d4-a716-446655440000"
study_setting_id = "550e8400-e29b-41d4-a716-446655440000"
study_setting_type = "General"
settings_file_path = r"c:\path\to\settings.json"
show_settings_example = True

study_add_configurations_resp:
list[CommandResponse[Contracts_SettingsChangedResponse]] =
pxc.study.add_configurations(study_id=study_id, study_setting_id=study_setting_id,
study_setting_type=study_setting_type, settings_file_path=settings_file_path,
show_settings_example=show_settings_example, print_message=True)
study_add_configurations_final: Contracts_SettingsChangedResponse =
SDKBase.get_response_data(study_add_configurations_resp)

if study_add_configurations_final is not None:
    # Iterate over StudyIds list
    if study_add_configurations_final.StudyIds is not None:
        for item in study_add_configurations_final.StudyIds:
            print(item)
    else:
        print(f"No StudyIds returned")
else:
    print(f"add_configurations failed: {study_add_configurations_resp.Message}")
```

Create Settings

Description: Execute create_settings operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study

- `study_setting_type` (str): study_setting_type parameter
- `settings_file_path` (str): settings_file_path parameter
- `show_settings_example` (bool): show_settings_example parameter (optional)

Response Structure:

- `StudyIds` (Optional[list[str]]): List of Studyids (list)

```
study_id = "550e8400-e29b-41d4-a716-446655440000"
study_setting_type = "General"
settings_file_path = r"c:\path\to\settings.json"
show_settings_example = True

study_create_settings_resp:
list[CommandResponse[Contracts_SettingsChangedResponse]] =
pxc.study.create_settings(study_id=study_id,
study_setting_type=study_setting_type, settings_file_path=settings_file_path,
show_settings_example=show_settings_example, print_message=True)
study_create_settings_final: Contracts_SettingsChangedResponse =
SDKBase.get_response_data(study_create_settings_resp)

if study_create_settings_final is not None:
    # Iterate over StudyIds list
    if study_create_settings_final.StudyIds is not None:
        for item in study_create_settings_final.StudyIds:
            print(item)
    else:
        print(f"No StudyIds returned")
else:
    print(f"create_settings failed: {study_create_settings_resp.Message}")
```

Delete Settings

Description: Execute delete_settings operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study
- `study_setting_id` (str): study_setting_id parameter

Response Structure:

- `StudyIds` (Optional[list[str]]): List of Studyids (list)

```
study_id = "550e8400-e29b-41d4-a716-446655440000"
study_setting_id = "550e8400-e29b-41d4-a716-446655440000"

study_delete_settings_resp:
list[CommandResponse[Contracts_SettingsChangedResponse]] =
pxc.study.delete_settings(study_id=study_id, study_setting_id=study_setting_id,
print_message=True)
study_delete_settings_final: Contracts_SettingsChangedResponse =
SDKBase.get_response_data(study_delete_settings_resp)

if study_delete_settings_final is not None:
    # Iterate over StudyIds list
    if study_delete_settings_final.StudyIds is not None:
        for item in study_delete_settings_final.StudyIds:
            print(item)
    else:
        print(f"No StudyIds returned")
else:
    print(f"delete_settings failed: {study_delete_settings_resp.Message}")
```

List Settings

Description: Execute list_settings operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study

Response Structure:

- `StudySettings` (Optional[list[Contracts_StudySetting]]): List of Studysettings (list)

  ▶ Properties of `Contracts_StudySetting`
    - `Name` (Optional[str]): Name value
    - `Id` (Optional[str]): Id value
    - `Status` (Optional[str]): Status value
    - `Type` (Optional[str]): Type value

```
study_id = "550e8400-e29b-41d4-a716-446655440000"

study_list_settings_resp:
list[CommandResponse[Contracts_ListStudySettingsResponse]] =
pxc.study.list_settings(study_id=study_id, print_message=True)
study_list_settings_final: Contracts_ListStudySettingsResponse =
SDKBase.get_response_data(study_list_settings_resp)

if study_list_settings_final is not None:
    # Iterate over StudySettings list
    if study_list_settings_final.StudySettings is not None:
        for item in study_list_settings_final.StudySettings:
            # Access properties of Contracts_StudySetting object
            print(f"Name: {item.Name}")
            print(f"Id: {item.Id}")
            print(f"Status: {item.Status}")
            print(f"Type: {item.Type}")
    else:
        print(f"No StudySettings returned")
```

```
else:
    print(f"list_settings failed: {study_list_settings_resp.Message}")
```

Download Specific Changeset

Description: Execute download_specific_changeset operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study
- `changeset_id` (uuid4()): Unique identifier for a specific changeset
- `output_directory_path` (str): output_directory_path parameter
- `list_files` (bool): list_files parameter (optional)

Response Structure:

- `StudyId` (Optional[str]): Studyid value (single)
- `ChangesetId` (Optional[str]): Changesetid value (single)
- `DownloadedFilePaths` (Optional[list[Contracts_StudyFile]]): List of Downloadedfilepaths (list)

  ▶ Properties of `Contracts_StudyFile`
  - `FilePath` (Optional[str]): Filepath value
  - `DataType` (Optional[str]): Datatype value

```
study_id = "550e8400-e29b-41d4-a716-446655440000"
changeset_id = "550e8400-e29b-41d4-a716-446655440000"
output_directory_path = r"c:\output"
list_files = True

study_download_specific_changeset_resp:
list[CommandResponse[Contracts_DownloadSpecificChangesetResponse]] =
pxc.study.download_specific_changeset(study_id=study_id,
changeset_id=changeset_id, output_directory_path=output_directory_path,
list_files=list_files, print_message=True)
study_download_specific_changeset_final:
Contracts_DownloadSpecificChangesetResponse =
SDKBase.get_response_data(study_download_specific_changeset_resp)

if study_download_specific_changeset_final is not None:
    # Iterate over DownloadedFilePaths list
    if study_download_specific_changeset_final.DownloadedFilePaths is not None:
        for item in study_download_specific_changeset_final.DownloadedFilePaths:
            # Access properties of Contracts_StudyFile object
            print(f"FilePath: {item.FilePath}")
            print(f"DataType: {item.DataType}")
    else:
        print(f"No DownloadedFilePaths returned")
    # Access single properties
    if study_download_specific_changeset_final.StudyId is not None:
        print(f"StudyId: {study_download_specific_changeset_final.StudyId}")
    if study_download_specific_changeset_final.ChangesetId is not None:
        print(f"ChangesetId:
{study_download_specific_changeset_final.ChangesetId}")
else:
    print(f"download_specific_changeset failed:
{study_download_specific_changeset_resp.Message}")
```

Get Changeset Sync Status

Description: Execute get_changeset_sync_status operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study

Response Structure:

- `Status` (Optional[Contracts_ChangesetSyncStatus]): Status value (single)

  ▶ Properties of `Contracts_ChangesetSyncStatus`
  - `InSync` (str): Insync value
  - `HasOutgoing` (str): Hasoutgoing value
  - `HasIncoming` (str): Hasincoming value

```
study_id = "550e8400-e29b-41d4-a716-446655440000"

study_get_changeset_sync_status_resp:
list[CommandResponse[Contracts_GetChangesetSyncStatusResponse]] =
pxc.study.get_changeset_sync_status(study_id=study_id, print_message=True)
study_get_changeset_sync_status_final: Contracts_GetChangesetSyncStatusResponse =
SDKBase.get_response_data(study_get_changeset_sync_status_resp)

if study_get_changeset_sync_status_final is not None:
    # Access single properties
    if study_get_changeset_sync_status_final.Status is not None:
        print(f"Status: {study_get_changeset_sync_status_final.Status}")
else:
    print(f"get_changeset_sync_status failed:
{study_get_changeset_sync_status_resp.Message}")
```

Get Last Changeset Id

Description: Execute get_last_changeset_id operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study

Response Structure:

- `ChangesetId` (Optional[str]): Changesetid value (single)

```
study_id = "550e8400-e29b-41d4-a716-446655440000"

study_get_last_changeset_id_resp:
list[CommandResponse[Contracts_GetLastChangesetIdResponse]] =
pxc.study.get_last_changeset_id(study_id=study_id, print_message=True)
study_get_last_changeset_id_final: Contracts_GetLastChangesetIdResponse =
SDKBase.get_response_data(study_get_last_changeset_id_resp)

if study_get_last_changeset_id_final is not None:
    # Access single properties
    if study_get_last_changeset_id_final.ChangesetId is not None:
        print(f"ChangesetId: {study_get_last_changeset_id_final.ChangesetId}")
else:
    print(f"get_last_changeset_id failed:
{study_get_last_changeset_id_resp.Message}")
```

Get Last Local Changeset Id

Description: Execute get_last_local_changeset_id operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study

Response Structure:

- `ChangesetId` (Optional[str]): Changesetid value (single)

```
study_id = "550e8400-e29b-41d4-a716-446655440000"

study_get_last_local_changeset_id_resp:
list[CommandResponse[Contracts_GetLastChangesetIdResponse]] =
pxc.study.get_last_local_changeset_id(study_id=study_id, print_message=True)
study_get_last_local_changeset_id_final: Contracts_GetLastChangesetIdResponse =
SDKBase.get_response_data(study_get_last_local_changeset_id_resp)

if study_get_last_local_changeset_id_final is not None:
    # Access single properties
    if study_get_last_local_changeset_id_final.ChangesetId is not None:
        print(f"ChangesetId:
{study_get_last_local_changeset_id_final.ChangesetId}")
else:
    print(f"get_last_local_changeset_id failed:
{study_get_last_local_changeset_id_resp.Message}")
```

Get Studies Download Urls

Description: Execute get_studies_download_urls operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study
- `changeset_id` (uuid4()): Unique identifier for a specific changeset

Response Structure:

- `StudyId` (Optional[str]): Studyid value (single)
- `ChangesetId` (Optional[str]): Changesetid value (single)
- `SimulationDataUrls` (Optional[list[Contracts_SimulationDataUrl]]): List of Simulationdataurls (list)

  ▶ Properties of `Contracts_SimulationDataUrl`
    - `Uri` (Optional[str]): Uri value
    - `Type` (Optional[str]): Type value

```
study_id = "550e8400-e29b-41d4-a716-446655440000"
changeset_id = "550e8400-e29b-41d4-a716-446655440000"

study_get_studies_download_urls_resp:
list[CommandResponse[Contracts_GetStudiesDownloadUrlsResponse]] =
pxc.study.get_studies_download_urls(study_id=study_id, changeset_id=changeset_id,
print_message=True)
study_get_studies_download_urls_final: Contracts_GetStudiesDownloadUrlsResponse =
SDKBase.get_response_data(study_get_studies_download_urls_resp)

if study_get_studies_download_urls_final is not None:
    # Iterate over SimulationDataUrls list
    if study_get_studies_download_urls_final.SimulationDataUrls is not None:
        for item in study_get_studies_download_urls_final.SimulationDataUrls:
            # Access properties of Contracts_SimulationDataUrl object
            print(f"Uri: {item.Uri}")
            print(f"Type: {item.Type}")
    else:
        print(f"No SimulationDataUrls returned")
    # Access single properties
    if study_get_studies_download_urls_final.StudyId is not None:
        print(f"StudyId: {study_get_studies_download_urls_final.StudyId}")
    if study_get_studies_download_urls_final.ChangesetId is not None:
        print(f"ChangesetId: {study_get_studies_download_urls_final.ChangesetId}")
else:
    print(f"get_studies_download_urls failed:
{study_get_studies_download_urls_resp.Message}")
```

List Changesets

Description: Execute list_changesets operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study

Response Structure:

- `StudyId` (Optional[str]): Studyid value (single)

- `Changesets` (Optional[list[Contracts_Changeset]]): List of Changesets (list)

  ▶ Properties of `Contracts_Changeset`
    - `Id` (Optional[GuidValue]): Id value
    - `CommitMessage` (Optional[str]): Commitmessage value
    - `LastUpdateMessage` (Optional[str]): Lastupdatemessage value
    - `CreatedByUserId` (Optional[StringValue]): Createdbyuserid value
    - `CreatedDate` (Optional[str]): Createddate value
    - `UpdatedDate` (Optional[str]): Updateddate value
    - `Status` (Optional[str]): Status value
    - `CreatedByUserName` (Optional[str]): Createdbyusername value

```
study_id = "550e8400-e29b-41d4-a716-446655440000"

study_list_changesets_resp:
list[CommandResponse[Contracts_ListChangesetsResponse]] =
pxc.study.list_changesets(study_id=study_id, print_message=True)
study_list_changesets_final: Contracts_ListChangesetsResponse =
SDKBase.get_response_data(study_list_changesets_resp)

if study_list_changesets_final is not None:
    # Iterate over Changesets list
    if study_list_changesets_final.Changesets is not None:
        for item in study_list_changesets_final.Changesets:
            # Access properties of Contracts_Changeset object
            print(f"Id: {item.Id}")
            print(f"CommitMessage: {item.CommitMessage}")
            print(f"LastUpdateMessage: {item.LastUpdateMessage}")
            print(f"CreatedByUserId: {item.CreatedByUserId}")
            print(f"CreatedDate: {item.CreatedDate}")
            # ... and 3 more properties
    else:
        print(f"No Changesets returned")
    # Access single properties
    if study_list_changesets_final.StudyId is not None:
        print(f"StudyId: {study_list_changesets_final.StudyId}")
else:
    print(f"list_changesets failed: {study_list_changesets_resp.Message}")
```

List Local Changesets

Description: Execute list_local_changesets operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study

Response Structure:

- `StudyId` (Optional[str]): Studyid value (single)

- `Changesets` (Optional[list[Contracts_Changeset]]): List of Changesets (list)

  ▶ Properties of `Contracts_Changeset`
    - `Id` (Optional[GuidValue]): Id value
    - `CommitMessage` (Optional[str]): Commitmessage value
    - `LastUpdateMessage` (Optional[str]): Lastupdatemessage value
    - `CreatedByUserId` (Optional[StringValue]): Createdbyuserid value
    - `CreatedDate` (Optional[str]): Createddate value
    - `UpdatedDate` (Optional[str]): Updateddate value
    - `Status` (Optional[str]): Status value
    - `CreatedByUserName` (Optional[str]): Createdbyusername value

```
study_id = "550e8400-e29b-41d4-a716-446655440000"

study_list_local_changesets_resp:
list[CommandResponse[Contracts_ListChangesetsResponse]] =
pxc.study.list_local_changesets(study_id=study_id, print_message=True)
study_list_local_changesets_final: Contracts_ListChangesetsResponse =
SDKBase.get_response_data(study_list_local_changesets_resp)

if study_list_local_changesets_final is not None:
    # Iterate over Changesets list
    if study_list_local_changesets_final.Changesets is not None:
        for item in study_list_local_changesets_final.Changesets:
            # Access properties of Contracts_Changeset object
            print(f"Id: {item.Id}")
            print(f"CommitMessage: {item.CommitMessage}")
            print(f"LastUpdateMessage: {item.LastUpdateMessage}")
            print(f"CreatedByUserId: {item.CreatedByUserId}")
            print(f"CreatedDate: {item.CreatedDate}")
            # ... and 3 more properties
    else:
        print(f"No Changesets returned")
    # Access single properties
    if study_list_local_changesets_final.StudyId is not None:
        print(f"StudyId: {study_list_local_changesets_final.StudyId}")
else:
    print(f"list_local_changesets failed:
{study_list_local_changesets_resp.Message}")
```

Pull Latest

Description: Execute pull_latest operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study

Response Structure:

- `StudyId` (Optional[str]): Studyid value (single)
- `CloudStudyName` (Optional[str]): Cloudstudyname value (single)
- `Success` (Optional[bool]): Success value (single)

```
study_id = "550e8400-e29b-41d4-a716-446655440000"

study_pull_latest_resp: list[CommandResponse[Contracts_PullLatestResponse]] =
pxc.study.pull_latest(study_id=study_id, print_message=True)
study_pull_latest_final: Contracts_PullLatestResponse =
SDKBase.get_response_data(study_pull_latest_resp)

if study_pull_latest_final is not None:
    # Access single properties
    if study_pull_latest_final.StudyId is not None:
        print(f"StudyId: {study_pull_latest_final.StudyId}")
    if study_pull_latest_final.CloudStudyName is not None:
        print(f"CloudStudyName: {study_pull_latest_final.CloudStudyName}")
    if study_pull_latest_final.Success is not None:
        print(f"Success: {study_pull_latest_final.Success}")
else:
    print(f"pull_latest failed: {study_pull_latest_resp.Message}")
```

Push Changeset

Description: Execute push_changeset operation

Parameters:

- `study_id` (uuid4()): Unique identifier for a specific study
- `commit_message` (str): commit_message parameter

Response Structure:

- `StudyId` (Optional[str]): Studyid value (single)
- `ChangesetId` (Optional[str]): Changesetid value (single)
- `CloudStudyName` (Optional[str]): Cloudstudyname value (single)

```
study_id = "550e8400-e29b-41d4-a716-446655440000"
commit_message = "Updated study configuration"

study_push_changeset_resp: list[CommandResponse[Contracts_PushChangesetResponse]]
= pxc.study.push_changeset(study_id=study_id, commit_message=commit_message,
print_message=True)
study_push_changeset_final: Contracts_PushChangesetResponse =
SDKBase.get_response_data(study_push_changeset_resp)

if study_push_changeset_final is not None:
    # Access single properties
    if study_push_changeset_final.StudyId is not None:
        print(f"StudyId: {study_push_changeset_final.StudyId}")
    if study_push_changeset_final.ChangesetId is not None:
        print(f"ChangesetId: {study_push_changeset_final.ChangesetId}")
    if study_push_changeset_final.CloudStudyName is not None:
        print(f"CloudStudyName: {study_push_changeset_final.CloudStudyName}")
else:
    print(f"push_changeset failed: {study_push_changeset_resp.Message}")
```