SAPManageProjectServices - New methods

Description

We are now going to create a number of “building block” methods, to allow all the component parts of a project update request to be constructed independently. This cleaner, more structured set of methods can then be used to replace the existing methods, as well as build in the additional functionality we’ll need going forward.

Method 1: ProjectUpdateRequest

Method signature

- Method name: ProjectUpdateRequest
- Return type: ProjectUpdateRequest
- Parameters:
    - ProjectReadByIDResponse project

Method details

Returns a new ProjectUpdateRequest object, with

- ProjectID set to the ProjectID of the project parameter

This is similar to how this is done in the current AddQmfIds method in SAPManageProjectServices.

Method 2: ProjectUpdateRequestTask

Method signature

- Method name: ProjectUpdateRequestTask
- Return type: ProjectUpdateRequestTask
- Parameters:
    - string taskId
    - string name
    - ProjectUpdateRequestTask.UUID uuid
    - ProjectUpdateRequestTask.ActionCode actionCode

Method details

Returns a new ProjectUpdateRequestTask object, with

- ProjectElementID.Value set to the taskId parameter
- Name.Name set to the name parameter

- UUID set to the uuid parameter
- ActionCode set to the actionCode parameter
- ActionCodeSpecified set to true

This is similar to how this is done in the current ProjectUpdateRequestTask constructor in Vision.Api.DotNet.ApplicationServices.SAPManageInvoice.SAPManageProjectIn.cs.

Method 3: ProjectUpdateRequestTaskExpense

Method signature

- Method name: ProjectUpdateRequestTaskExpense
- Return type: ProjectUpdateRequestTaskExpense
- Parameters:
    - string generalLedgerAccountAliasCode
    - decimal Value
    - string currencyCode
    - ProjectUpdateRequestTaskExpense.UUID uuid
    - ProjectUpdateRequestTask.ActionCode actionCode

Method details

Returns a new ProjectUpdateRequestTaskExpense object, with

- GeneralLedgerAccountAliasCode.Value set to the value parameter
- PlannedCostsAmount.Value set to the value parameter
- PlannedCostsAmount.currencyCode set to the currencyCode parameter
- UUID set to the uuid parameter
- ActionCode set to the actionCode parameter
- ActionCodeSpecified set to truee

This should be similar to how this is done in the commented-out code in Vision.Api.DotNet.ApplicationServices.SAPManageInvoice.SAPManageProjectIn.cs.

Method 4: ProjectUpdateRequestTaskRevenue

Method signature

- Method name: ProjectUpdateRequestTaskRevenue
- Return type: ProjectUpdateRequestTaskRevenue
- Parameters:
    - string generalLedgerAccountAliasCode
    - decimal Value
    - string currencyCode
    - ProjectUpdateRequestTaskRevenue.UUID uuid
    - ProjectUpdateRequestTask.ActionCode actionCode

Method details

Returns a new ProjectUpdateRequestTaskRevenue object, with

- GeneralLedgerAccountAliasCode.Value set to the value parameter
- PlannedCostsAmount.Value set to the value parameter
- PlannedCostsAmount.currencyCode set to the currencyCode parameter
- UUID set to the uuid parameter
- ActionCode set to the actionCode parameter
- ActionCodeSpecified set to true

This should be similar to how this is done in the commented-out code in Vision.Api.DotNet.ApplicationServices.SAPManageInvoice.SAPManageProjectIn.cs.

Method 5: AddTasksToProjectUpdateRequest

Method signature

- Method name: AddTasksToProjectUpdateRequest
- Return type: ProjectUpdateRequest
- Parameters:
    - ProjectUpdateRequest projectUpdateRequest
    - ProjectUpdateRequestTask[] tasks

Method details

Adds the supplied tasks to the supplied ProjectUpdateRequest, then returns the updated ProjectUpdateRequest object.

- ProjectUpdateRequest Task property is set to the supplied tasks parameter
- ProjectUpdateRequest TaskListCompleteTransmissionIndicator property is set to true

This is similar to how this is done in the current AddQmfIds method in SAPManageProjectServices.

Method 6: AddExpensesToProjectTaskUpdate

Method signature

- Method name: AddExpensesToProjectTaskUpdate
- Return type: ProjectUpdateRequestTask
- Parameters:
    - ProjectUpdateRequestTask projectUpdateRequestTask
    - ProjectUpdateRequestTaskExpense[] expenses

Method details

Adds the supplied expenses to the supplied ProjectUpdateRequestTask, then returns the updated ProjectUpdateRequestTask object.

- ProjectUpdateRequestTask Expense property is set to the supplied expenses parameter

This should be similar to how this is done in the commented-out code in Vision.Api.DotNet.ApplicationServices.SAPManageInvoice.SAPManageProjectIn.cs.

Method 7: AddRevenuesToProjectTaskUpdate

Method signature

- Method name: AddRevenuesToProjectTaskUpdate
- Return type: ProjectUpdateRequestTask
- Parameters:
  - ProjectUpdateRequestTask projectUpdateRequestTask
  - ProjectUpdateRequestTaskRevenue[] revenues

Method details

Adds the supplied revenues to the supplied ProjectUpdateRequestTask, then returns the updated ProjectUpdateRequestTask object.

- ProjectUpdateRequestTask Revenue property is set to the supplied revenues parameter

This should be similar to how this is done in the commented-out code in Vision.Api.DotNet.ApplicationServices.SAPManageInvoice.SAPManageProjectIn.cs.

Method 8: SendProjectUpdateRequest

Method signature

- Method name: SendProjectUpdateRequest
- Return type: ProjectUpdateConfirmationMessage_sync
- Parameters:
  - ProjectUpdateRequest projectUpdateRequest

Method details

Sends the supplied ProjectUpdateRequest, then returns the response object.

This is similar to how this is done in the current AddQmfIds method in SAPManageProjectServices; from "var response…", but ***without*** *the last section that checks the counts*

The following two are ones we haven't tested out yet … I ***think*** these should be the correct parameters based on the WSDL, but when you get around to testing we may need to tweak some of these … so ***possibly*** a little more R&D work required

Method 9: ProjectReleaseTaskReleaseRequest

Method signature

- Method name: ProjectReleaseTaskReleaseRequest
- Return type: ProjectReleaseTaskReleaseRequest
- Parameters:
    - ProjectUpdateRequest.UUID projectUuid
    - string projectId
    - ProjectUpdateRequestTask.UUID taskUuid

Method details

Returns a new ProjectReleaseTaskReleaseRequest used to release an existing project task

- ProjectID property is set to the supplied projectId parameter
- UUID property is set to the supplied projectUuid parameter (***note: this is an assumption!***)
- Task is a new ProjectReleaseTaskReleaseRequestTask object
    - UUID property is set to the supplied taskUuid parameter (***note: this is an assumption!***)

Method 10: SendProjectReleaseTaskReleaseRequest

Method signature

- Method name: SendProjectReleaseTaskReleaseRequest
- Return type: ProjectReleaseTaskReleaseConfirmationMessage_sync
- Parameters:
    - ProjectReleaseTaskReleaseRequest projectReleaseTaskReleaseRequest

Method details

Sends the supplied ProjectReleaseTaskReleaseRequest, then returns the response object.

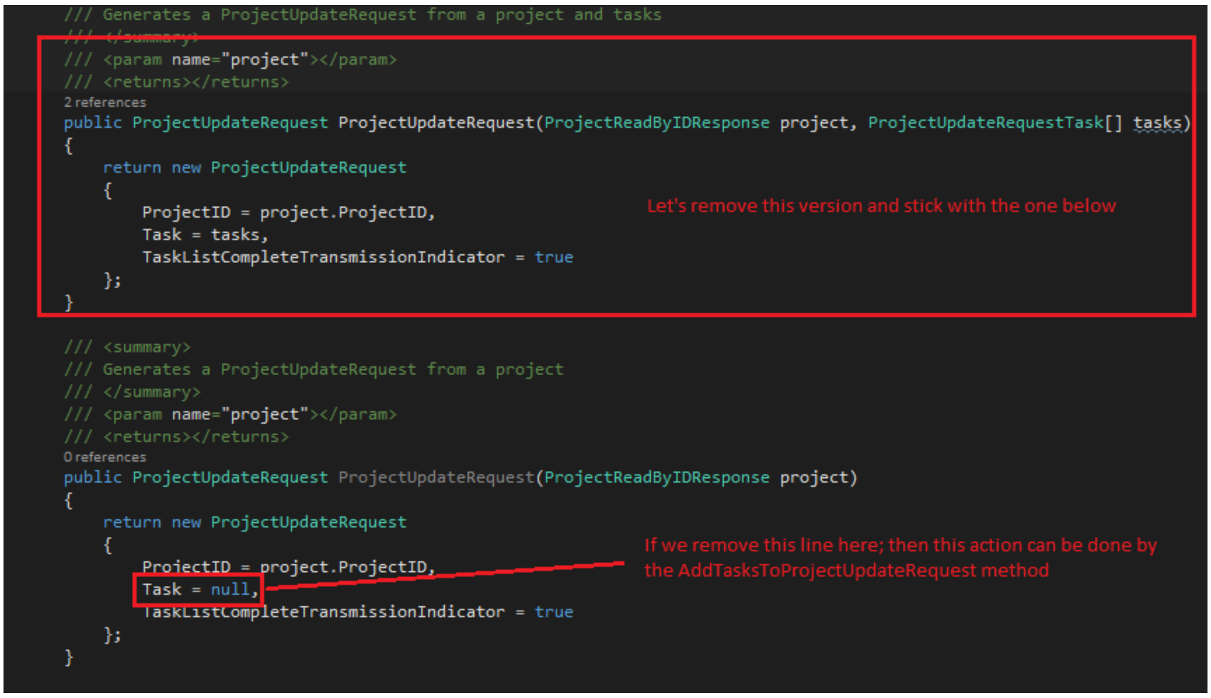
(note: assumptions here! but based on how SendProjectUpdateRequest works)

Code Review Feedback

Hi (X),

These methods are looking really good! Just a few comments below if you could take a look when you get chance…

Method 1 - ProjectUpdateRequest



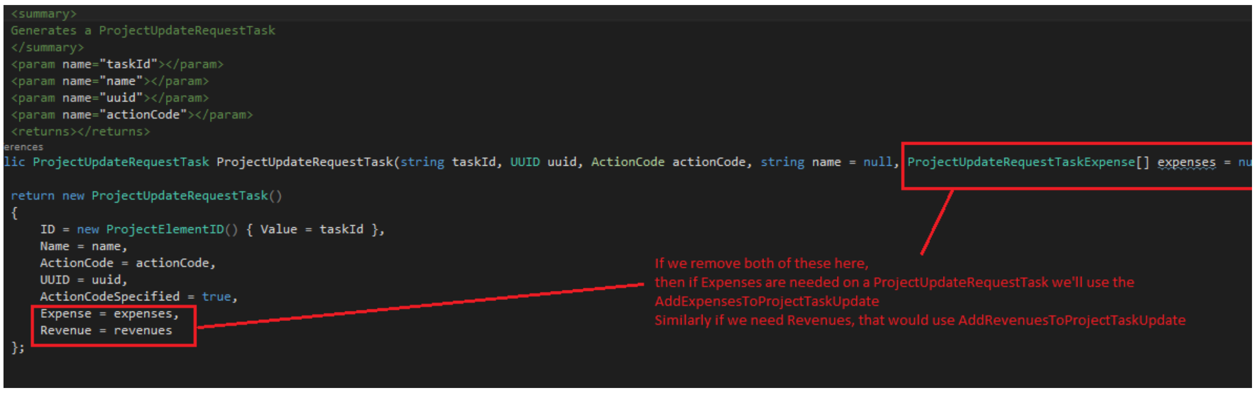
```
/// Generates a ProjectUpdateRequest from a project and tasks
/// </summary>
/// <param name="project"></param>
/// <returns></returns>
2 references
public ProjectUpdateRequest ProjectUpdateRequest(ProjectReadByIDResponse project, ProjectUpdateRequestTask[] tasks)
{
    return new ProjectUpdateRequest
    {
        ProjectID = project.ProjectID,
        Task = tasks,
        TaskListCompleteTransmissionIndicator = true
    };
}

/// <summary>
/// Generates a ProjectUpdateRequest from a project
/// </summary>
/// <param name="project"></param>
/// <returns></returns>
0 references
public ProjectUpdateRequest ProjectUpdateRequest(ProjectReadByIDResponse project)
{
    return new ProjectUpdateRequest
    {
        ProjectID = project.ProjectID,
        Task = null,
        TaskListCompleteTransmissionIndicator = true
    };
}
```

Let's remove this version and stick with the one below

If we remove this line here; then this action can be done by the AddTasksToProjectUpdateRequest method

Method 2 - ProjectUpdateRequestTask

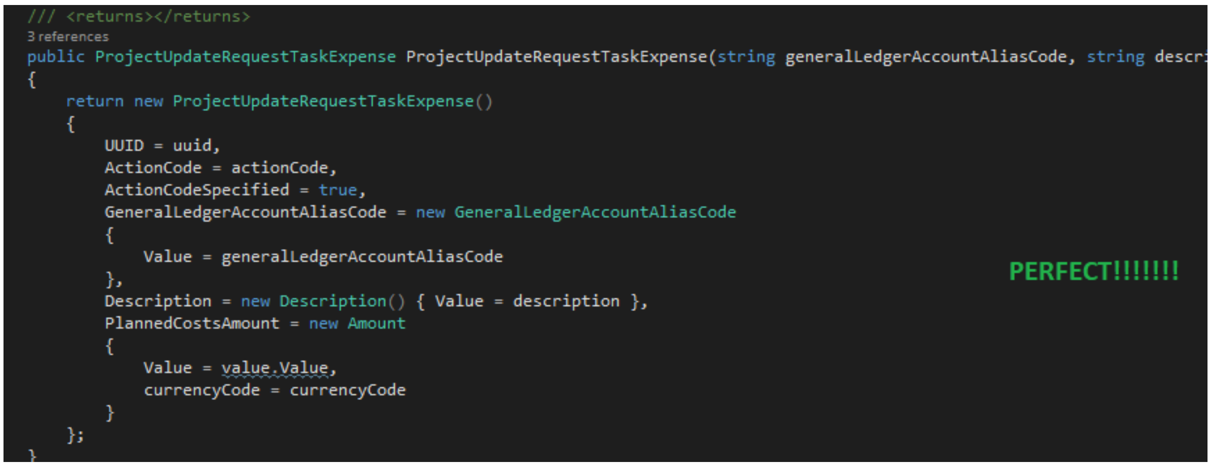


```
<summary>
Generates a ProjectUpdateRequestTask
</summary>
<param name="taskId"></param>
<param name="name"></param>
<param name="uuid"></param>
<param name="actionCode"></param>
<returns></returns>
erences
lic ProjectUpdateRequestTask ProjectUpdateRequestTask(string taskId, UUID uuid, ActionCode actionCode, string name = null, ProjectUpdateRequestTaskExpense[] expenses = nu

return new ProjectUpdateRequestTask()
{
    ID = new ProjectElementID() { Value = taskId },
    Name = name,
    ActionCode = actionCode,
    UUID = uuid,
    ActionCodeSpecified = true,
    Expense = expenses,
    Revenue = revenues
};
```

If we remove both of these here,
then if Expenses are needed on a ProjectUpdateRequestTask we'll use the AddExpensesToProjectTaskUpdate
Similarly if we need Revenues, that would use AddRevenuesToProjectTaskUpdate

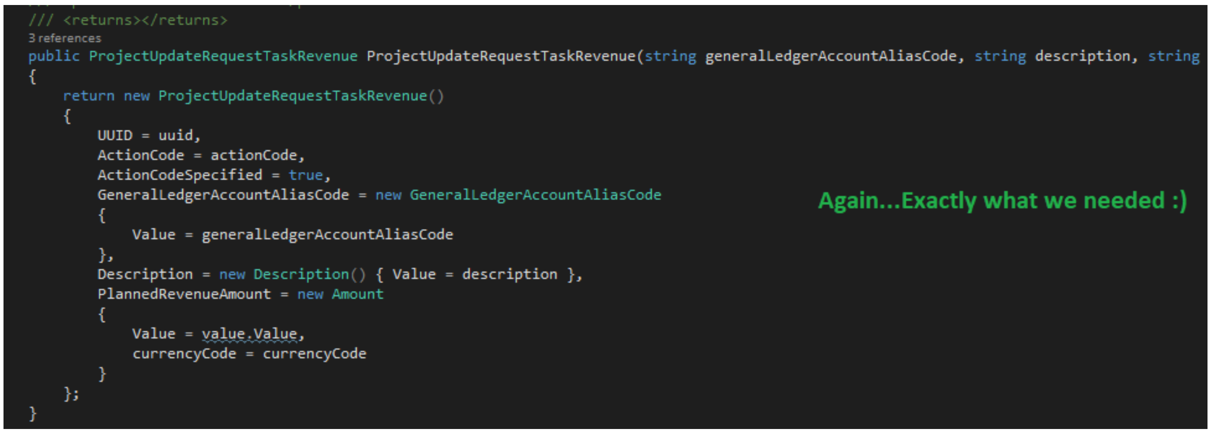
Method 3 - ProjectUpdateRequestTaskExpense



```
/// <returns></returns>
3 references
public ProjectUpdateRequestTaskExpense ProjectUpdateRequestTaskExpense(string generalLedgerAccountAliasCode, string descr
{
    return new ProjectUpdateRequestTaskExpense()
    {
        UUID = uuid,
        ActionCode = actionCode,
        ActionCodeSpecified = true,
        GeneralLedgerAccountAliasCode = new GeneralLedgerAccountAliasCode
        {
            Value = generalLedgerAccountAliasCode
        },
        Description = new Description() { Value = description },
        PlannedCostsAmount = new Amount
        {
            Value = value.Value,
            currencyCode = currencyCode
        }
    };
}
```

PERFECT!!!!!!!

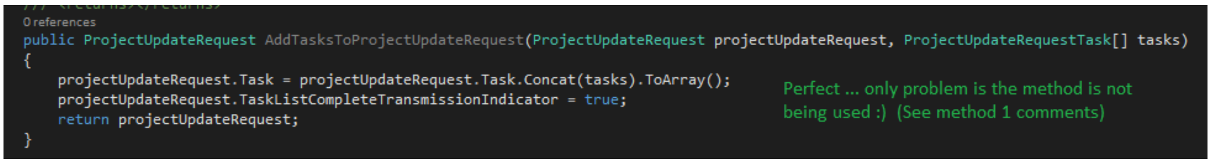
Method 4 - ProjectUpdateRequestTaskRevenue



```
/// <returns></returns>
3 references
public ProjectUpdateRequestTaskRevenue ProjectUpdateRequestTaskRevenue(string generalLedgerAccountAliasCode, string description, string
{
    return new ProjectUpdateRequestTaskRevenue()
    {
        UUID = uuid,
        ActionCode = actionCode,
        ActionCodeSpecified = true,
        GeneralLedgerAccountAliasCode = new GeneralLedgerAccountAliasCode
        {
            Value = generalLedgerAccountAliasCode
        },
        Description = new Description() { Value = description },
        PlannedRevenueAmount = new Amount
        {
            Value = value.Value,
            currencyCode = currencyCode
        }
    };
}
```

Again...Exactly what we needed :)

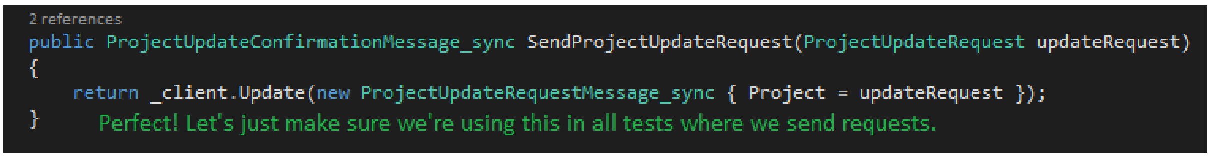
Method 5 - AddTasksToProjectUpdateRequest



```
0 references
public ProjectUpdateRequest AddTasksToProjectUpdateRequest(ProjectUpdateRequest projectUpdateRequest, ProjectUpdateRequestTask[] tasks)
{
    projectUpdateRequest.Task = projectUpdateRequest.Task.Concat(tasks).ToArray();
    projectUpdateRequest.TaskListCompleteTransmissionIndicator = true;
    return projectUpdateRequest;
}
```

Perfect ... only problem is the method is not being used :) (See method 1 comments)

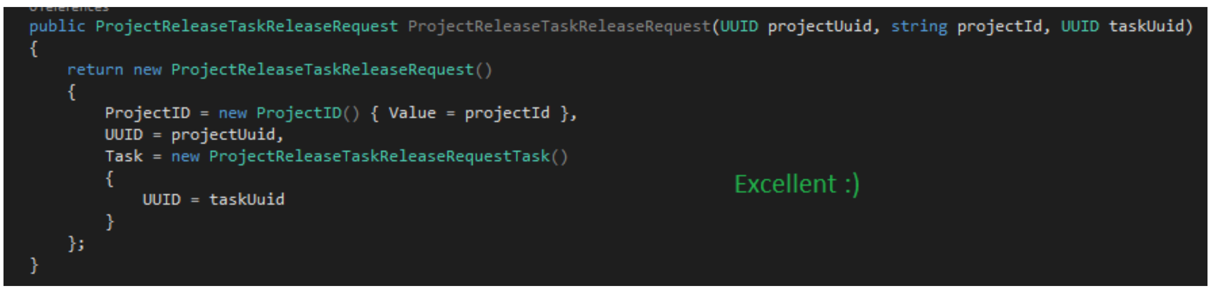
Method 8 - SendProjectUpdateRequest



```
2 references
public ProjectUpdateConfirmationMessage_sync SendProjectUpdateRequest(ProjectUpdateRequest updateRequest)
{
    return _client.Update(new ProjectUpdateRequestMessage_sync { Project = updateRequest });
}
```

Perfect! Let's just make sure we're using this in all tests where we send requests.

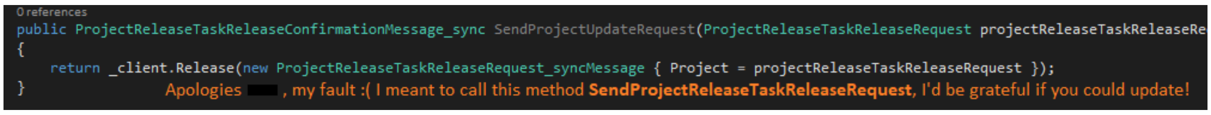
Method 9 - ProjectReleaseTaskReleaseRequest



```
public ProjectReleaseTaskReleaseRequest ProjectReleaseTaskReleaseRequest(UUID projectUuid, string projectId, UUID taskUuid)
{
    return new ProjectReleaseTaskReleaseRequest()
    {
        ProjectID = new ProjectID() { Value = projectId },
        UUID = projectUuid,
        Task = new ProjectReleaseTaskReleaseRequestTask()
        {
            UUID = taskUuid
        }
    };
}
```

Excellent :)

Method 10 - SendProjectReleaseTaskReleaseRequest



```
0 references
public ProjectReleaseTaskReleaseConfirmationMessage_sync SendProjectUpdateRequest(ProjectReleaseTaskReleaseRequest projectReleaseTaskReleaseRe
{
    return _client.Release(new ProjectReleaseTaskReleaseRequest_syncMessage { Project = projectReleaseTaskReleaseRequest });
}
```

Apologies [redacted], my fault :(I meant to call this method **SendProjectReleaseTaskReleaseRequest**, I'd be grateful if you could update!