Проекты VK WorkSpace

Описание скриптовой автоматизации

Портал с документацией: https://biz.mail.ru/docs/on-premises/

Оглавление

Общая информация

Скрипт — это набор инструкций на C#, которые составляют тело скрытого внутреннего метода, затем компилируемого и выполняемого по триггерам.

Технически скрипты транслируются на внутренний промежуточный язык .NET - CIL.

Скрипты вызываются событиями в системе. На одно событие в рамках пространства может быть создано несколько автоматизаций.

Контекст событий

Каждое событие имеет свой контекст - набор данных, связанный с ним. Контексты для разных событий отличаются. Контекст передаётся в память выполняющегося скрипта в виде типизированного объекта. Доступ к контексту осуществляется через переменную `e`.

```
debug($"Пространство: {e.WorkspaceKey}");
```

Контекст создания задачи

Когда триггером автоматизации является событие Создание задачи, передаются следующие данные события:

```
public class WorkitemCreatedEventContext : BaseWorkitemEventContext
{
    public Guid WorkitemId { get; init; } // Идентификатор созданной задачи
    public string WorkitemKey { get; init; } // Ключ созданной задачи
    public string WorkitemName { get; init; } // Название созданной задачи
    public required string? WorkitemDescription { get; set; } // Описание созданной задачи
    public Guid? WorkitemAssigneeId { get; init; } // Идентификатор ответсвенного на задаче
    public Guid WorkitemTypeId { get; init; } // Идентификатор типа задачи
    public required DateTime WorkitemCreationDate { get; set; } // Дата создания задачи
    public required Guid? WorkitemWorkflowId { get;
init; } // Идентификатор рабочего процесса задачи

    public Guid? TenantId { get; init; } // Идентификатор тенанта
    public Guid EventId { get; init; } // Идентификатор события
    public EventTypes EventType { get; init; } // Тип события
    public string EventName { get; init; } // Название события
    public Guid UserId { get; init; } // Идентификатор пользователя, инициировавшего событие
    public Guid WorkspaceId { get; init; } // Идентификатор пространства, в котором произошло
событие
    public string WorkspaceKey { get; init; } // Ключ пространства, в котором произошло
событие
}
```

Контекст событий изменения задачи

Событиями изменения задачи являются:

- Изменение родительской задачи или папки;
- Изменение типа задачи;
- Изменение процесса;
- Изменение статуса;
- Изменение имени;
- Изменение описания;
- Изменение ответственного;
- Изменение даты начала;
- Изменение даты выполнения;
- Изменение оценки в часах;
- Изменение оценки в сторипоинтах;
- Изменение спринта;
- Изменение значения прогресса;
- Изменение пользовательского атрибута.

Для этих событий передаются следующие данные контекста:

```
public class WorkitemUpdatedEventContext : BaseWorkitemEventContext
{
    public object? OldValue { get; init; } // Значение параметра до изменения
    public object? NewValue { get; init; } // Значение параметра после изменения

    public Guid WorkitemId { get; init; } // Идентификатор задачи
    public string WorkitemKey { get; init; } // Ключ задачи
    public string WorkitemName { get; init; } // Название задачи
    public required string? WorkitemDescription { get; set; } // Описание задачи
    public Guid? WorkitemAssigneeId { get; init; } // Идентификатор ответственного на задаче
    public Guid WorkitemTypeId { get; init; } // Идентификатор типа задачи
    public required DateTime WorkitemCreationDate { get; set; } // Дата создания задачи
    public required Guid? WorkitemWorkflowId { get;
init; } // Идентификатор рабочего процесса задачи

    public Guid WorkitemParentId { get; init; } // Идентификатор родительского элемента
(задачи или папки)
    public Guid? WorkitemStatusId { get; init; } // Идентификатор статуса
    public Guid WorkitemCreatedBy { get; set; } // Идентификатор пользователя, который создал
задачу
    public Guid WorkitemUpdatedBy { get; set; } // Идентификатор пользователя, который
последний изменил задачу
    public DateTime WorkitemUpdateDate { get; set; } // Дата изменения задачи
    public DateTime? WorkitemDueDate { get; set; } // Дата выполнения
    public DateTime? WorkitemEndDate { get; set; } // Дата завершения
    public DateTime? WorkitemStartDate { get; set; } // Дата начала
    public Dictionary<Guid, object?> WorkitemAttributesValues { get; set; } = new(); //
Значения пользовательских атрибутов
    public OkrProgressType? WorkitemProgressType { get; set; }
    public string? WorkitemProgressData { get; set; }
```

```
    public int? WorkitemProgressValue { get; set; }
    public int? WorkitemEstimatedTime { get; set; } // Оценка в часах
    public int? WorkitemSpentTime { get; set; } // Сумма затраченного времени
    public int? WorkitemLeftTime { get; set; } // Оставшееся время
    public int? WorkitemEstimatedStoryPoints { get; set; } // Оценка в сторипоинтах

    public Guid? TenantId { get; init; } // Идентификатор тенанта
    public Guid EventId { get; init; } // Идентификатор события
    public EventTypes EventType { get; init; } // Тип события
    public string EventName { get; init; } // Название события
    public Guid UserId { get; init; } // Идентификатор пользователя, инициировавшего событие
    public Guid WorkspaceId { get; init; } // Идентификатор пространства, в котором произошло
событие
    public string WorkspaceKey { get; init; } // Ключ пространства, в котором произошло
событие
}
```

Соответствие типа объекта в полях `NewValue` и `OldValue` в зависимости от типа события приведено в таблице:

Событие	Тип объекта	Описание объекта
Изменение родительской задачи или папки	`Guid`	Идентификатор задачи или папки
Изменение типа задачи	`Guid?`	Идентификатор типа
Изменение процесса	`Guid?`	Идентификатор процесса
Изменение статуса	`Guid`	Идентификатор статуса
Изменение имени	`string`	Название задачи
Изменение описания	`string`	Описание задачи
Изменение ответственного	`Guid?`	Идентификатор ответственного
Изменение даты начала	`DateTime?`	Дата начала
Изменение даты выполнения	`DateTime?`	Дата выполнения
Изменение оценки в часах	`int?`	Оценка в секундах
Изменение оценки в сторипоинтах	`int?`	Оценка в сторипоинтах

Событие	Тип объекта	Описание объекта
Изменение спринта	`Guid?`	Идентификатор спринта
Изменение значения прогресса	`int?`	Значение прогресса в %
Изменение пользовательского атрибута "Число"	`int?`	Числовое значение атрибута
Изменение пользовательского атрибута "Строка"	`int?`	Строковое значение атрибута
Изменение пользовательского атрибута "Выбор из списка"	`string?`	Значение опции атрибута
Изменение пользовательского атрибута "Тег"	`string?`	Значение опции атрибута
Изменение пользовательского атрибута "Пользователь"	`Guid?`	Идентификатор пользователя
Изменение пользовательского атрибута "Дата"	`DateTime?`	Значение даты атрибута
Изменение пользовательского атрибута "Время"	`int?`	Значение времени в секундах

Контекст создания списания

Когда триггером автоматизации является событие *Добавление списания времени", передаются следующие данные события:

```
public class TimeTrackingCreatedEventContext : EventContext
{
    public Guid WorkitemId { get; set; } // Идентификатор задачи по которой добавлено списание
    public Guid EntryId { get; set; } // Идентификатор списания
    public DateTime CreationDate { get; set; } // Дата добавления списания
    public DateTime StartDate { get; set; } // Дата списания
    public int Duration { get; set; } // Время списания в секундах
    public string? Description { get; set; } // Описание списания

    public Guid? TenantId { get; init; } // Идентификатор тенанта
    public Guid EventId { get; init; } // Идентификатор события
    public EventTypes EventType { get; init; } // Тип события
    public string EventName { get; init; } // Название события
    public Guid UserId { get; init; } // Идентификатор пользователя, инициировавшего событие
    public Guid WorkspaceId { get; init; } // Идентификатор пространства, в котором произошло
событие
```

```
    public string WorkspaceKey { get; init; } // Ключ пространства, в котором произошло
событие
}
```

Контекст изменения списания

Когда триггером автоматизации является событие "Редактирование списания времени" передаются следующие данные события:

```
public class TimeTrackingUpdatedEventContext : EventContext
{
    public Guid WorkitemId { get; set; } // Идентификатор задачи по которой добавлено списание
    public Guid EntryId { get; set; } // Идентификатор списания
    public DateTime CreationDate { get; set; } // Дата добавления списания
    public DateTime StartDate { get; set; } // Дата списания
    public int Duration { get; set; } // Время списания в секундах
    public string? Description { get; set; } // Описание списания

    public Guid? TenantId { get; init; } // Идентификатор тенанта
    public Guid EventId { get; init; } // Идентификатор события
    public EventTypes EventType { get; init; } // Тип события
    public string EventName { get; init; } // Название события
    public Guid UserId { get; init; } // Идентификатор пользователя, инициировавшего событие
    public Guid WorkspaceId { get; init; } // Идентификатор пространства, в котором произошло
событие
    public string WorkspaceKey { get; init; } // Ключ пространства, в котором произошло
событие
}
```

Контекст добавления задачи в элемент портфеля

Когда триггером автоматизации является событие Добавление задачи в элемент портфеля, передаются следующие данные события:

```
public class WorkitemAddedToPortfolioElementEventContext : EventContext
{
    public Guid WorkitemId { get; set; } // Идентификатор задачи, добавленной в элемент
портфеля
    public Guid PortfolioElementId { get; set; } // Идентификатор элемента портфеля
    public string PortfolioElementName { get; set; } = default!; // Название элемента портфеля
    public Guid? PortfolioElementStatusId { get; set; } // Идентификатор статуса элемента
портфеля

    public Guid? TenantId { get; init; } // Идентификатор тенанта
    public Guid EventId { get; init; } // Идентификатор события
    public EventTypes EventType { get; init; } // Тип события
    public string EventName { get; init; } // Название события
    public Guid UserId { get; init; } // Идентификатор пользователя, инициировавшего событие
    public Guid WorkspaceId { get; init; } // Идентификатор пространства, в котором произошло
событие
    public string WorkspaceKey { get; init; } // Ключ пространства, в котором произошло
событие
}
```

Контекст удаления задачи из элемента портфеля

Когда триггером автоматизации является событие Удаление задачи из элемента портфеля передаются следующие данные события:

```
public class WorkitemDeletedFromPortfolioElementEventContext : EventContext
{
    public Guid WorkitemId { get; set; } // Идентификатор задачи, добавленной в элемент
портфеля
    public Guid PortfolioElementId { get; set; } // Идентификатор элемента портфеля
    public string PortfolioElementName { get; set; } = default!; // Название элемента портфеля
    public Guid? PortfolioElementStatusId { get; set; } // Идентификатор статуса элемента
портфеля

    public Guid? TenantId { get; init; } // Идентификатор тенанта
    public Guid EventId { get; init; } // Идентификатор события
    public EventTypes EventType { get; init; } // Тип события
    public string EventName { get; init; } // Название события
    public Guid UserId { get; init; } // Идентификатор пользователя, инициировавшего событие
    public Guid WorkspaceId { get; init; } // Идентификатор пространства, в котором произошло
событие
    public string WorkspaceKey { get; init; } // Ключ пространства, в котором произошло
событие
}
```

Контекст создания спринта

Когда триггером автоматизации является событие Создание спринта, передаются следующие данные события:

```
public class SprintCreatedEventContext : EventContext
{
    public Guid ExtensionId { get; set; } // Идентификатор расширения Agile
    public Guid SprintId { get; set; } // Идентификатор спринта
    public string SprintName { get; set; } // Название спринта
    public SprintStatus SprintStatus { get; set; } // Статус спринта

    public Guid? TenantId { get; init; } // Идентификатор тенанта
    public Guid EventId { get; init; } // Идентификатор события
    public EventTypes EventType { get; init; } // Тип события
    public string EventName { get; init; } // Название события
    public Guid UserId { get; init; } // Идентификатор пользователя, инициировавшего событие
    public Guid WorkspaceId { get; init; } // Идентификатор пространства, в котором произошло
событие
    public string WorkspaceKey { get; init; } // Ключ пространства, в котором произошло
событие
}
```

Контекст изменения статуса спринта

Когда триггером автоматизации является событие Изменение статуса спринта, передаются следующие данные события:

```
public class SprintUpdatedEventContext : EventContext
{
    public Guid ExtensionId { get; set; } // Идентификатор расширения Agile
    public Guid SprintId { get; set; } // Идентификатор спринта
    public string SprintName { get; set; } // Название спринта
    public SprintStatus SprintStatus { get; set; } // Статус спринта

    public Guid? TenantId { get; init; } // Идентификатор тенанта
    public Guid EventId { get; init; } // Идентификатор события
    public EventTypes EventType { get; init; } // Тип события
    public string EventName { get; init; } // Название события
    public Guid UserId { get; init; } // Идентификатор пользователя, инициировавшего событие
    public Guid WorkspaceId { get; init; } // Идентификатор пространства, в котором произошло
событие
    public string WorkspaceKey { get; init; } // Ключ пространства, в котором произошло
событие
}
```

Контекст создания и блокировки страницы

Когда триггером автоматизации являются события Создание страницы, Блокирование страницы, Разблокирование страницы передаются следующие данные события:

```
public class DocumentCreatedEventContext : BaseDocumentEventContext
{
    public Guid DocumentId { get; init; } // Идентификатор страницы
    public required string DocumentKey { get; init; } // Ключ страницы
    public required string DocumentName { get; init; } // Название страницы
    public string? Content { get; init; } // Содержимое страницы
    public int Version { get; init; } // Номер версии страницы
    public required DateTime CreationDate { get; set; } // Дата создания страницф
    public required string DocumentUrl { get; init; } // Ссылка на страницу
    public required string VersionUrl { get; init; } // Ссылка на версию страницы
    public List<Guid>? MentionedUserIds { get; init; } // Идентификаторы пользователей,
упомянутых на странице

    public Guid? TenantId { get; init; } // Идентификатор тенанта
    public Guid EventId { get; init; } // Идентификатор события
    public EventTypes EventType { get; init; } // Тип события
    public string EventName { get; init; } // Название события
    public Guid UserId { get; init; } // Идентификатор пользователя, инициировавшего событие
    public Guid WorkspaceId { get; init; } // Идентификатор пространства, в котором произошло
событие
    public string WorkspaceKey { get; init; } // Ключ пространства, в котором произошло
событие
}
```

Контекст изменения страницы

События изменения страницы являются:

- Изменение названия страницы;
- Изменение содержания страницы;

- Изменение статуса страницы;
- Изменение меток страницы;

Для этих событий передаются следующие данные контекста:

```
public class DocumentUpdatedEventContext : BaseDocumentEventContext
{
    public DateTime UpdateDate { get; set; } // Время изменения страницы
    public object? OldValue { get; init; } // Значение параметра до изменения
    public object? NewValue { get; init; } // Значение параметра после изменения

    public Guid DocumentId { get; init; } // Идентификатор страницы
    public required string DocumentKey { get; init; } // Ключ страницы
    public required string DocumentName { get; init; } // Название страницы
    public string? Content { get; init; } // Содержимое страницы
    public int Version { get; init; } // Номер версии страницы
    public required DateTime CreationDate { get; set; } // Дата создания страницф
    public required string DocumentUrl { get; init; } // Ссылка на страницу
    public required string VersionUrl { get; init; } // Ссылка на версию страницы
    public List<Guid>? MentionedUserIds { get; init; } // Идентификаторы пользователей,
упомянутых на странице

    public Guid? TenantId { get; init; } // Идентификатор тенанта
    public Guid EventId { get; init; } // Идентификатор события
    public EventTypes EventType { get; init; } // Тип события
    public string EventName { get; init; } // Название события
    public Guid UserId { get; init; } // Идентификатор пользователя, инициировавшего событие
    public Guid WorkspaceId { get; init; } // Идентификатор пространства, в котором произошло
событие
    public string WorkspaceKey { get; init; } // Ключ пространства, в котором произошло
событие
}
```

Соответствие типа объекта в полях `NewValue` и `OldValue` в зависимости от типа события приведено в таблице:

Событие	Тип объекта	Описание объекта
Изменение названия страницы	`string`	Название страницы
Изменение содержания страницы	`string`	Содержимое страницы
Изменение статуса страницы	`Guid`	Идентификатор статуса
Изменение меток страницы	`List<string>`	Список названий меток

Контекст удаления страницы

Когда триггером автоматизации является событие Удаление страницы, передаются следующие данные события:

```
public class DocumentDeletedEventContext : BaseDocumentEventContext
{
    public Guid DocumentId { get; init; } // Идентификатор страницы

    public Guid? TenantId { get; init; } // Идентификатор тенанта
    public Guid EventId { get; init; } // Идентификатор события
    public EventTypes EventType { get; init; } // Тип события
    public string EventName { get; init; } // Название события
    public Guid UserId { get; init; } // Идентификатор пользователя, инициировавшего событие
    public Guid WorkspaceId { get; init; } // Идентификатор пространства, в котором произошло
событие
    public string WorkspaceKey { get; init; } // Ключ пространства, в котором произошло
событие
}
```

Методы Script API

Метод, в котором выполняется пользовательский код, технически находится в классе, наследуемом от базового класса, имеющим веб-клиент `CwmPublicAPI` и набор методов.

Примеры использования функций приведены в подразделе Примеры.

Сигнатуры методов

Сигнатуры методов имеют своими аргументами идентификаторы или имена/ключи задач, пространств, папок и т.д., там где это применимо, но они, как правило, необязательные. Если что-либо пропущено, то подразумевается текущее пространство, текущая задача и т.п., а для удобства использования они перечислены в порядке возрастания иерархии:

`GetWorkitem(string? workitemKey = null, string? workspaceKey = null);`

Методы работы с пространствами

```
Task<List<WorkspaceModel>?> GetWorkspaces()
Task<WorkspaceModel?> GetWorkspace(Guid workspaceId)
Task<WorkspaceModel?> GetWorkspace(string workspaceKey = null)

Task<WorkspaceModel> CreateWorkspace(string name, string key, string? description = null)

Task<WorkspaceModel> UpdateWorkspaceName(string name, string? workspaceKey = null)

Task DeleteWorkspace(string? workspaceKey = null)
```

Методы работы с ролями

```
Task<List<RoleModel>?> GetRoles(string? workspaceKey = null)
Task<List<RoleModel>?> GetRoles(bool? isSystem, string? workspaceKey = null)
Task<RoleModel?> GetRole(Guid roleId, string? workspaceKey = null)
Task<RoleModel?> GetRole(string role, string? workspaceKey = null)

Task<RoleModel> CreateRole(string name, List<Permission?>? permissions, string? workspaceKey
= null)

Task<RoleModel> UpdateRolePermissions(Guid roleId, List<Permission?>? permissions = null,
string? workspaceKey = null)
```

Методы работы с пользователями

```
Task<List<ProviderModel>?> GetProviders()

Task<List<UserModel>?> GetUsers()
Task<UserModel?> GetUser(Guid userId)
Task<UserModel?> GetUser(string userName, Guid? providerId = null)

Task<List<GroupModel>?> GetUsersGroups(string userName)

Task<List<UserModel>?> GetUsersByRole(Guid roleId, string? workspaceKey = null)
Task<List<UserModel>?> GetUsersByRole(string roleDisplayName, string? workspaceKey = null)
Task AddUserRole(Guid userId, Guid roleId, string? workspaceKey = null)
Task DeleteUserRole(Guid userId, Guid roleId, string? workspaceKey = null)
```

Методы работы с группами

```
Task<List<GroupModel>?> GetUsersGroups()
Task<GroupModel?> GetUserGroup(Guid userGroupId)
Task<GroupModel?> GetUserGroup(string name, Guid? providerId = null)

Task DeleteGroupRole(Guid groupId, Guid roleId, string? workspaceKey = null)
```

Методы работы с папками

```
Task<List<FolderModel>?> GetFolders(string? workspaceKey = null)
Task<List<FolderModel>?> GetFolderByParent(Guid parentId, string? workspaceKey = null)
Task<FolderModel?> GetFolder(Guid folderId, string? workspaceKey = null)
Task<FolderModel?> GetFolder(string name, string? workspaceKey = null)

Task<FolderModel> CreateFolder(string name, Guid? parentId = null, string? workspaceKey =
null)

Task<FolderModel?> UpdateFolder(PatchFolderRequestBody body, Guid folderId, string?
workspaceKey = null) // общего вида - сразу все свойства скопом
Task<WorkitemModel> UpdateFolderName(string name, Guid folderId, string? workspaceKey = null)

Task DeleteFolder(Guid folderId, string? workspaceKey = null)
```

Методы работы с типами

```
Task<List<TypeModel>?> GetTypes(string? workspaceKey = null)
Task<TypeModel?> GetType(Guid typeId, string? workspaceKey = null)
Task<TypeModel?> GetType(string typeName, string? workspaceKey = null)

Task<TypeModel> CreateType(CreateTypeRequestBody body, string? workspaceKey = null)

Task DeleteType(Guid typeId, string? workspaceKey = null)
Task DeleteType(string? type, string? workspaceKey = null)
```

Методы работы со статусами

```
Task<List<StatusCategoryModel>?> GetStatusCategories()

Task<List<StatusModel>?> GetStatuses(string? workspaceKey = null)
Task<StatusModel?> GetStatus(Guid statusId, string? workspaceKey = null)
Task<StatusModel?> GetStatus(string name, string? workspaceKey = null)

Task<StatusModel> CreateStatus(CreateStatusRequestBody body, string? workspaceKey = null)
```

Методы работы с атрибутами типов

```
Task<List<AttributeModel>?> GetAttributes(string? workspaceKey = null)
Task<AttributeModel?> GetAttribute(Guid attributeId, string? workspaceKey = null)
Task<AttributeModel?> GetAttribute(string attributeName, string? workspaceKey = null)

Task<AttributeModel> CreateNumberAttribute(string name, string? description, string?
workspaceKey = null)
Task<AttributeModel> CreateUniStringAttribute(string name, string? description = null,
string? workspaceKey = null)
Task<AttributeModel> CreateUserAttribute(string name, string? description = null, string?
workspaceKey = null)
Task<AttributeModel> CreateDateAttribute(string name, string? description = null, string?
workspaceKey = null)
Task<AttributeModel> CreateTimeAttribute( string name, string? description = null, string?
workspaceKey = null)
Task<AttributeModel> CreateUniSelectAttribute(string name, IEnumerable<string> listOptions,
string? description = null, string? workspaceKey = null)
Task<AttributeModel> CreateTagAttribute(string name, LIEnumerablest<string> listOptions,
string? description = null, string? workspaceKey = null)

Task<AttributeModel> UpdateWorkitemAttribute(Guid attributeId, PatchAttributeRequestBody
body, string? workspaceKey = null)
Task<AttributeModel> UpdateScalarWorkitemAttribute(Guid attributeId, string name, string?
description = null, string? workspaceKey = null) // Все скалярные
Task<AttributeModel> UpdateListWorkitemAttribute(Guid attributeId, string name,
List<PatchAttributeOptionModel> listOptions, string? description = null, string? workspaceKey
= null) // UniSelect & Tag

Task DeleteAttribute(Guid attributeId, string? workspaceKey = null)
```

Методы работы с задачами

```
Task<List<WorkitemModel>?> GetWorkitems(string? workspaceKey = null)
Task<WorkitemModel?> GetWorkitem(Guid workitemId, string? workspaceKey = null)
Task<WorkitemModel?> GetWorkitem(string? workitemKey = null, string? workspaceKey = null)
Task<List<WorkitemModel>?> GetWorkitemsByParent(bool withSubitems = false, string? parentKey
= null, string? workspaceKey =
null) // планарный список для иерархии потомков (withSubitems = true)
Task<WorkitemModelTree?> GetWorkitemTree(Guid? parentId = null, string? workspaceKey = null)

Task<WorkitemModel> CreateWorkitem(CreateWorkitemRequestBody body, string? workspaceKey =
null)

Task<WorkitemModel?> UpdateWorkitem(PatchWorkitemRequestBody body, string? workitemKey =
null, string? workspaceKey = null) // общего вида
Task<WorkitemModel> UpdateWorkitemParent(Guid? parentId, string? workitemKey = null, string?
workspaceKey = null)
Task<WorkitemModel> UpdateWorkitemStatus(string status, string? workitemKey = null, string?
workspaceKey = null)
Task<WorkitemModel> UpdateWorkitemType(string typeName, string? workitemKey = null, string?
workspaceKey = null)
Task<WorkitemModel> UpdateWorkitemName(string name, string? workitemKey = null, string?
workspaceKey = null)
Task<WorkitemModel> UpdateWorkitemDescription(string desc, string? workitemKey = null,
string? workspaceKey = null)
Task<WorkitemModel> UpdateWorkitemAssignee(Guid userId, string? workitemKey = null, string?
workspaceKey = null)
Task<WorkitemModel> UpdateWorkitemAssignee(string? userName, string? workitemKey = null,
string? workspaceKey = null)
Task<WorkitemModel> UpdateWorkitemStartDate(DateTime? startDate, string? workitemKey = null,
string? workspaceKey = null)
Task<WorkitemModel> UpdateWorkitemDueDate(DateTime? dueDate, string? workitemKey = null,
string? workspaceKey = null)
Task<WorkitemModel> UpdateWorkitemSprint(Guid? sprintId, string? workitemKey = null, string?
workspaceKey = null)
Task<WorkitemModel> UpdateWorkitemStoryPoints(int? points, string? workitemKey = null,
string? workspaceKey = null)
Task<WorkitemModel> UpdateWorkitemOriginalEstimate(int? estimate, string? workitemKey = null,
string? workspaceKey = null)
Task<WorkitemModel> UpdateWorkitemWorkflow(Guid? workflowId, string? workitemKey = null,
string? workspaceKey = null)
Task<WorkitemModel> UpdateWorkitemPortfolioElement(List<Guid?>? elementIds, string?
workitemKey = null, string? workspaceKey = null)

Task<List<SharedWorkitemPermissionModel>> GetWorkitemSharing(string? workitemKey = null,
string? workspaceKey = null)
Task<SharedWorkitemPermissionModel?> ShareWorkitemForUser(SharedItemAccessLevel accessLevel,
Guid userId, string? workitemKey = null, string? workspaceKey = null)
Task<SharedWorkitemPermissionModel?> ShareWorkitemForGroup(SharedItemAccessLevel accessLevel,
Guid groupId, string? workitemKey = null, string? workspaceKey = null)
Task<SharedWorkitemPermissionModel?> UpdateWorkitemSharing(Guid permissionId,
SharedItemAccessLevel accessLevel, string? workitemKey = null, string? workspaceKey = null)
Task DeleteWorkitemSharing(Guid permissionId, string? workitemKey = null, string?
workspaceKey = null)

Task DeleteWorkitem(string? workitemKey = null, string? workspaceKey = null)
```

Методы работы со значениями атрибутов

Атрибуты с типом Число

```
Task<NumberFieldValueModel?> GetNumberAttributeValue(Guid attributeId, string? workitemKey =
null, string? workspaceKey = null)
Task<NumberFieldValueModel?> GetNumberAttributeValue(string attributeName, string?
workitemKey = null, string? workspaceKey = null)

Task<NumberFieldValueModel?> UpdateNumberAttributeValue(string attributeName, double? value,
string? workitemKey = null, string? workspaceKey = null)
Task<NumberFieldValueModel?> UpdateNumberAttributeValue(Guid attributeId, double? value,
string? workitemKey = null, string? workspaceKey = null)
```

Атрибуты с типом Строка

```
Task<UniStringFieldValueModel?> GetUniStringAttributeValue(Guid attributeId, string?
workitemKey = null, string? workspaceKey = null)
Task<UniStringFieldValueModel?> GetUniStringAttributeValue(string attributeName, string?
workitemKey = null, string? workspaceKey = null)

Task<UniStringFieldValueModel?> UpdateUniStringAttributeValue(string attributeName, string
value, string? workitemKey = null, string? workspaceKey = null)
Task<UniStringFieldValueModel?> UpdateUniStringAttributeValue(Guid attributeId, string value,
string? workitemKey = null, string? workspaceKey = null)
```

Атрибуты с типом Выбор из списка

```
Task<UniSelectFieldValueModel?> GetUniSelectAttributeValue(Guid attributeId, string?
workitemKey = null, string? workspaceKey = null)
Task<UniSelectFieldValueModel?> GetUniSelectAttributeValue(string attributeName, string?
workitemKey = null, string? workspaceKey = null)

Task<UniSelectFieldValueModel?> UpdateUniSelectAttributeValue(string attributeName, string
option, string? workitemKey = null, string? workspaceKey = null)
Task<UniSelectFieldValueModel?> UpdateUniSelectAttributeValue(Guid attributeId, string
option, string? workitemKey = null, string? workspaceKey = null)
```

Атрибуты с типом Тег

```
Task<TagFieldValueModel?> GetTagAttributeValue(Guid attributeId, string? workitemKey = null,
string? workspaceKey = null)
Task<TagFieldValueModel?> GetTagAttributeValue(string attributeName, string? workitemKey =
null, string? workspaceKey = null)

Task<TagFieldValueModel?> UpdateTagAttributeValue(string attributeName, List<string> options,
string? workitemKey = null, string? workspaceKey = null)
Task<TagFieldValueModel?> UpdateTagAttributeValue(Guid attributeId, List<string> options,
string? workitemKey = null, string? workspaceKey = null)
```

Атрибуты с типом Дата

```
Task<DateFieldValueModel?> GetDateAttributeValue(Guid attributeId, string? workitemKey =
null, string? workspaceKey = null)
Task<DateFieldValueModel?> GetDateAttributeValue(string attributeName, string? workitemKey =
null, string? workspaceKey = null)

Task<DateFieldValueModel?> UpdateDateAttributeValue(string attributeName, DateTime? value,
string? workitemKey = null, string? workspaceKey = null)
Task<DateFieldValueModel?> UpdateDateAttributeValue(Guid attributeId, DateTime? value,
string? workitemKey = null, string? workspaceKey = null)
```

Атрибуты с типом Пользователь

```
Task<UserFieldValueModel?> GetUserAttributeValue(Guid attributeId, string? workitemKey =
null, string? workspaceKey = null)
Task<UserFieldValueModel?> GetUserAttributeValue(string attributeName, string? workitemKey =
null, string? workspaceKey = null)

Task<UserFieldValueModel?> UpdateUserAttributeValue(string attributeName, Guid userId,
string? workitemKey = null, string? workspaceKey = null)
Task<UserFieldValueModel?> UpdateUserAttributeValue(Guid attributeId, Guid userId, string?
workitemKey = null, string? workspaceKey = null)
```

Атрибуты с типом Время

```
Task<TimeFieldValueModel?> GetTimeAttributeValue(Guid attributeId, string? workitemKey =
null, string? workspaceKey = null)
Task<TimeFieldValueModel?> GetTimeAttributeValue(string attributeName, string? workitemKey =
null, string? workspaceKey = null)

Task<TimeFieldValueModel?> UpdateTimeAttributeValue(string attributeName, int? seconds,
string? workitemKey = null, string? workspaceKey = null)
Task<TimeFieldValueModel?> UpdateTimeAttributeValue(Guid attributeId, int? seconds, string?
workitemKey = null, string? workspaceKey = null)
```

Методы работы с комментариями

```
Task<List<CommentModel>?> GetWorkitemComments(string? workitemKey = null, string?
workspaceKey = null)

Task<CommentModel> CreateWorkitemComment(string text, string? workitemKey = null, string?
workspaceKey = null)
```

Методы работы с вложениями

```
Task<List<AttachmentModel>?> GetWorkitemAttachments(string? workitemKey = null, string?
workspaceKey = null)
```

```
Task<AttachmentModel?> GetWorkitemAttachment(Guid attachmentId, string? workitemKey = null,
string? workspaceKey = null)

Task DeleteAttachment(Guid attachmentId, string? workitemKey = null, string? workspaceKey =
null)
```

Методы работы с портфелями

```
Task<List<PortfolioModel>?> GetPortfolios(string? workspaceKey = null)
Task<List<PortfolioModel>?> GetPortfoliosInFolder(Guid folderId, string? workspaceKey = null)
Task<PortfolioModel?> GetPortfolio(Guid portfolioId, string? workspaceKey = null)
Task<PortfolioModel?> GetPortfolio(string name, string? workspaceKey = null)

Task<PortfolioModel> CreatePortfolio(string name, Guid folderId, string? workspaceKey = null)

Task<PortfolioModel> UpdatePortfolio(Guid portfolioId, string name, string? workspaceKey =
null)

Task DeletePortfolio(Guid portfolioId, string? workspaceKey = null)
```

Работа с элементами портфеля

```
Task<List<PortfolioElementModel>?> GetPortfolioElements(string? workspaceKey = null)
Task<List<PortfolioElementModel>?> GetPortfolioElementsInFolder(Guid folderId, string?
workspaceKey = null)
Task<List<PortfolioElementModel>?> GetPortfolioElementsForStatus(string status, string?
workspaceKey = null)
Task<List<PortfolioElementModel>?> GetPortfolioElements(Guid portfolioId, string?
workspaceKey = null)
Task<PortfolioElementModel?> GetPortfolioElement(Guid elementId, string? workspaceKey = null)
Task<PortfolioElementModel?> GetPortfolioElement(string elementName, string? workspaceKey =
null)

Task<PortfolioElementModel> CreatePortfolioElement(CreatePortfolioElementRequestBody body,
string? workspaceKey = null)

Task<PortfolioElementModel> UpdatePortfolioElement(PortfolioElementModel body, string?
workspaceKey = null)

Task DeletePortfolioElement(Guid elementId, string? workspaceKey = null)
```

Получение задач в элементе портфеля

```
Task<List<WorkitemModel>?> GetWorkitemsByPortfolioElement(Guid elementId, string?
workspaceKey = null)
```

Методы работы с Agile

```
Task<List<AgileModel>?> GetAgileExtensions(string? workspaceKey = null)
Task<AgileModel?> GetAgileExtension(Guid agileExtensionId, string? workspaceKey = null)
```

```
Task<AgileModel> CreateAgile(CreateAgileRequestBody body, string? workspaceKey = null)

Task DeleteAgile(Guid agileExtensionId, string? workspaceKey = null)
```

Работа со спринтами

```
Task<List<SprintModel>?> GetSprints(string? workspaceKey = null)
Task<SprintModel?> GetSprint(Guid sprintId, string? workspaceKey = null)
Task<SprintModel?> GetSprint(string name, string? workspaceKey = null)

Task<SprintModel> CreateSprint(CreateSprintRequestBody body, string? workspaceKey = null)

Task DeleteSprint(Guid sprintId, string? workspaceKey = null)
```

Методы работы со списаниями

```
Task<List<TimeTrackingEntryModel>?> GetTimeTrackingEntries(DateTime? startDate = null,
DateTime? endDate = null,
    List<string>? users = null, int? maxItemsCount = 50, string? fromToken = null)
```

Методы работы со связями

```
Task<List<LinkTypeModel>?> GetLinkTypes(string? workspaceKey = null)

Task<List<WorkitemLinkModel>?> GetWorkitemLinks(string? workitemKey = null, string?
workspaceKey = null)
Task<WorkitemLinkModel?> CreateWorkitemLink(string linkedWorkitemKey, string type, string?
workitemKey = null, string? workspaceKey = null)
Task DeleteWorkitemLink(Guid linkId, string? workitemKey = null, string? workspaceKey = null)

Task<List<WorkitemModel>?> GetDocumentLinkedWorkitems(Guid? documentId = null, string?
workspaceKey = null)
Task<List<DocumentModel>?> GetWorkitemLinkedDocuments(string? workitemKey = null, string?
workspaceKey = null)
Task AddWorkitemAndDocumentLink(Guid linkedDocumentId, string? documentWorkspaceKey = null,
string? workitemKey = null, string? workitemWorkspaceKey = null)
Task DeleteWorkitemAndDocumentLink(Guid linkedDocumentId, string? documentWorkspaceKey =
null, string? workitemKey = null, string? workitemWorkspaceKey = null)
```

Методы работы со страницами

```
Task<List<DocumentModel>> GetDocuments(string? workspaceKey = null)
Task<DocumentModel> GetDocument(string? documentKey = null, string? workspaceKey = null)

Task<List<DocumentStatusModel>?> GetDocumentStatuses(string? workspaceKey = null)
Task<DocumentStatusModel?> GetDocumentStatus(Guid statusId, string? workspaceKey = null)
Task<DocumentStatusModel?> GetDocumentStatus(string statusName, string? workspaceKey = null)
Task<DocumentModel> UpdateDocumentStatus(string status, Guid? documentId = null, string?
```

```
workspaceKey = null)

Task<DocumentModel?> BlockDocument(Guid? documentId = null, string? workspaceKey = null)
Task<DocumentModel?> UnblockDocument(Guid? documentId = null, string? workspaceKey = null)

Task<List<SharedDocumentPermissionModel>> GetDocumentSharing(string? documentKey = null,
string? workspaceKey = null)
Task<SharedDocumentPermissionModel> ShareDocumentForUser(SharedItemAccessLevel accessLevel,
Guid userId, string? documentKey = null, string? workspaceKey = null)
Task<SharedDocumentPermissionModel> ShareDocumentForGroup(SharedItemAccessLevel accessLevel,
Guid groupId, string? documentKey = null, string? workspaceKey = null)
Task<SharedDocumentPermissionModel?> UpdateDocumentSharing(Guid permissionId,
SharedItemAccessLevel accessLevel, string? documentKey = null, string? workspaceKey = null)
Task DeleteDocumentSharing(Guid permissionId, string? documentKey = null, string?
workspaceKey = null)
```

Методы для отправки HTTP-запросов

```
Task<string> SendHttpRequest(string url, HttpRequest request) // аргумент request содержит в
себе все возможные свойства, передаваемые в следующей функции напрямую

Task<string> SendHttpRequest(string url, HttpMethod method = HttpMethod.Get, object? body =
null,
    List<(string Name, string Value)>? headers = null, string mediaType = "application/json",
    IReadOnlyDictionary<string, object?>? options = null)
```

Примеры

> **Примечание**
>
> Для всех функций ScriptAPI (доступ к PublicAPI и логирование) необходимо указывать впереди ключевое слово await, так как они асинхронные.

Синтаксис языка — C11, поэтому инициализация коллекций происходит следующим образом: `List<int> list = new { 1, 2 }`, а не `List<int> list = [ 1, 2 ]`.

Простейший пример скрипта, который даже не использует PublicAPI:

```
debug("USER = " + e.UserId);
```

> **Примечание**
>
> Если тело скрипта передается в запросе напрямую (на странице Swagger или утилите типа Postman), все внутренние двойные кавычки свойства, передаваемого JSON, необходимо экранировать:

```
{
   Name: "Script1",
   ScriptType: "Method",
```

```
   EventType: "WorkitemNameChanged",
   Script: "debug(\"USER = \" + e.UserId);", // вот тут в тексте самого скрипта -
экранирование кавычек
}
```

Пример создания папки со случайным именем

```
var r = new Random();
var newName = $"name{r.Next(0, 1000)}";
await CreateFolder(newName); // вот тут должно быть await
debug("New name = " + newName);
```

Примеры создания скриптов

```
// Сравнение с текущим событием можно будет делать как по его Id, так и по его имени (enum
лучше)
// Кейс 1: e.EventId == Events.WorkitemCreated (Events - класс с полями в виде именованных
guid + некоторые вспомогательные функции)
// Кейс 2: e.EventType == EventTypes.WorkitemCreated (EventTypes - enum)
// Events также имеет статический словарь '<Guid, EventTypes> Map' - Id и enum события для
системных полей
// Имена обсуждаемы

// Сценарий: автозаполнение исполнителя в зависимости от типа запроса
var wiType = await GetType(e.WorkitemTypeId);
if (e.EventType == EventTypes.WorkitemCreated && wiType.Name == "Bug") {
    // UserModel? assignee = await GetUser("ivan.ivanov"); // в данном случае получение модели
излишне
    WorkitemModel? workitem = await UpdateWorkitemAssignee("ivan.ivanov");
}

//  Сценарий: по умолчанию проставление среднего приоритета при регистрации запроса
var wiType = await GetType(e.WorkitemTypeId);
if (e.EventId == EventTypes.WorkitemCreated && wiType.Name == "Bug") {
    // если уже имеется Id атрибута, лучше использовать функцию, принимающую именно его, а не
имя (производительность)
    await UpdateUniSelectAttributeValue("Приоритет", "Средний"); // текущий воркайтем
    //await SetUniSelectAttributeValue(attributeId, "Средний", workitemId2); // сторонний
воркайтем, но текущего воркспейса
    //await SetUniSelectAttributeValue(attributeId, "Средний", workitemId2, workspaceId2); //
воркайтем в стороннем воркспейсе
}

// Сценарий: проставление команды в зависимости от исполнителя
var user = await GetUser(e.WorkitemAssigneeId);
// 'WorkitemChanged' нет - все события на изменение гранулярные
if (e.EventType == EventTypes.WorkitemNameChanged && user.Username == "ivan.ivanov") {
    await UpdateUniSelectAttributeValue("Команда", "Alfa");
}

// Сценарий: Автоматическое назначение на задачу определенной проектной роли. Как следствие
назначается пользователь, у которого установлена такая проектная роль.
List<UserModel> users = await GetUsersByRole(roleId); // можно так (все функции планарно на
одном уровне, но функции могут быть длиннее по имени - с префиксом сущности)
// List<UserModel> users = await WorkspaceUsers.FindByRole(role); // а можно эдак (в АПИ
```

```
организовать отдельные 'свойства доступа': для текущего воркайтема, общего доступа в
воркспейсам, юзерам и т.д.)

// Сценарий: Автоматическое присвоение типа документа по вхождению определенного текста в
название документа.
// Сценарий: Создание задачи с автоматическим определением типа на базе локации пользователя,
то есть где пользователь инициировал создание задачи. Чаще всего применимо к спискам
(портфелям).
WorkitemModel wi = await UpdateWorkitemType("SomeType"); // текущему воркайтему

// Сценарий: Расчет и заполнение атрибута в зависимости от значений других атрибутов текущей
задачи. Например, автоматическое присваивание задаче портфеля, если она является определенного
типа.
PortfolioElementModel? elem1 = await GetPortfolioElement(elementId);
PortfolioElementModel? elem1 = await GetPortfolioElement("Element1"); // или по elementName
// PorfolioElementModel? elem1 = await PorfolioElements.FindByName(name); // либо так
elem1.StartDate = DateTimeOffset.UtcNow.AddDays(10).Date; // изменить сам элемент
await UpdatePortfolioElement(elem1);
await UpdateWorkitemPortfolioElements(elementIds); // привязка к воркайтему списка элементов
портфеля - по guid

// Сценарий: Установить ответственного при определённых значениях
var priorityValue = await GetUniSelectAttributeValue(priorityAttributeName);
var tagValue = await GetTagAttributeValue(tagAttributeName); // текущий воркайтем
if (priorityValue.Value.Name == "Высокий" && tagValue.Value.Any(a => a.Name == "UX")) {
    await UpdateWorkitemAssignee(userId);  // текущему воркайтему!
}

// Автоматически рассчитать стоимость задачи и записать результат в атрибут "Общая стоимость",
перемножив значения атрибутов "Оценка в часах" и "Стоимость часа".
NumberFieldValueModel attr1 = await GetNumberAttributeValue("Оценка в часах");
NumberFieldValueModel attr2 = await GetNumberAttributeValue("Стоимость часа");
if(attr1.Value != null && attr2.Value != null)
{
    var val3 = attr1.Value * attr2.Value;
    await UpdateNumberAttributeValue("Общая стоимость", val3);
}

// Автоматически рассчитать сумму атрибутов “Оценка” во всех дочерних задачах и заполнить
рассчитанным значением атрибута “Оценка“ в родительской задаче.
var tree = await GetWorkitemTree();
List<WorkitemModel> sons = tree.GetChildren();
var result = 0.0;
foreach (var son in sons)
{
    var attribute = son.Attributes.Single(a => a.NumberFieldValueModel.Name == "Оценка");
    result += attribute.NumberFieldValueModel.Value ?? 0;
}

await UpdateNumberAttributeValue("Оценка", result);
```

Технический писатель: Белова Ирина

6 мая 2026 г.