

Table of Contents

Part I Stimulsoft Server	1
1 .NET API	1
Users	1
Signing Up	2
Logging In	3
Getting User Info	3
Getting List of Users	5
Changing User Info	6
Changing User Password	6
Creating New User	7
Deleting User	8
Logging Out	9
Roles	10
Creating and Saving Roles	11
Getting Role Info	12
Getting List of Roles	13
Deleting Role	14
Workspaces	15
Creating and Saving	15
Getting Workspace Info	16
Getting List of Workspaces	17
Deleting Workspace	18
Items	19
Creating and Saving Items	19
Sharing Items	20
Getting Item	21
Getting List of Items	22
Deleting Item	23
Resource Item	24
Attaching Items to Reports and Files	26
Running Report	27
Exporting Report Snapshot	28
Scheduling Actions	30
Scheduling Frequency	33
Data Sources	37
File	39
2 REST API	41
Object Model	45
Sign Up	47
Login	48
Logout	49
Users	50
GET List	51
GET Info	53
POST Create	54
PUT Edit	55
DELETE	56
Change password	56
Reset password	57

Roles	58
GET List	61
GET Info	63
POST Create	64
PUT Edit	65
DELETE	66
Items	67
GET List	70
GET Info	73
GET Versions	74
POST Create	75
PUT Edit	76
PUT Restore	77
DELETE	78
Share	79
GET Info	80
PUT Edit	81
DELETE Reset	82
Items Attach	82
Items Detach	83
Scheduler	84
GET List	86
GET Info	87
POST Create	89
PUT Edit	90
DELETE	91
GET_Info (Status)	92
PUT Edit (Status)	93
Run	94
Files	94
GET List	97
GET Download	98
POST Create	100
PUT Append	101
DELETE	103
Report Template	103
GET List	105
GET Info	106
POST Create	107
DELETE	108
Run	109
POST Duplicate	111
Report Snapshot	113
GET List	114
GET Info	115
POST Create	116
DELETE	117
Export	118
Export	120
PDF	122
XPS	125
Power Point	128
HTML	131
Text	134

Rich Text 137

Word 139

Open Document Writer..... 142

Excel 145

Open Document Calc..... 148

Data 151

Image 154

Download 157

Track Task Status 159

Index **0**

1 Stimulsoft Server

This documentation part contains instructions to access the functions of Stimulsoft Server. **Stimulsoft Server** is a complete business intelligence, client-server solution that provides reporting and analytics:

- [.NET API](#) provides quick and easy access to the main functions of the system and automates user actions of Stimulsoft Server in your application.
- [REST API](#) allows you to get access to the main functions of Stimulsoft Server by REST from any application written on any language.

Information

You can download the package on the [project page](#) or install it directly from the NuGet package console:

```
PM> Install-Package  
Stimulsoft_Reports.Server_Connect_API
```

1.1 .NET API

The **Stimulsoft Server** platform provides a full-featured backend solution for your .NET application. API provides quick and easy access to the main functions of the system and automates user actions of Stimulsoft Server in your application. You can use the asynchronous or synchronous methods of some root classes that implement work with basic objects (Users, Roles and Items) of Stimulsoft Server.

1.1.1 Users

To access information about users of the **Stimulsoft Server**, there is a specialized user class called **StiUserConnection** that automatically handles much of the functionality required for user account management. With this class, you will be able to add user account functionality in your app. In order to use the class **StiUserConnection** you should create an instance of **StiServerConnection** and call one of its methods (**StiServerConnection.Accounts.Users**). Access to information about the user provides the **StiUser** class. The user name or the key can obtain an instance of this class. Use for this methods **GetByName()** (**GetByNameAsync()**) and **GetByKey()** (**GetByKeyAsync()**) respectively. There is also fastest way to get an instance of the current user by calling the **Current** property in the

StiServerConnection.Accounts.Users class.

1.1.1.1 Signing Up

To create a new user, you must use the method **SignUp()** or asynchronous version **SignUpAsync()**:

.NET API

```
...
public void SignUp()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.SignUp("UserName", "Password");
}
...
```

Or asynchronous method:

.NET API

```
...
public async void SignUpAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.SignUpAsync("UserName", "Password");
}
...
```

This call will create a new user in your instance of Stimulsoft Server. Before it does this, it also checks to make sure that both the username and email are unique. The password (up to 6 characters) is stored on the server as a hash and never sent to the client in the form of a plaintext.

In addition, we have two overridden methods for the signup action with additional info (first name and last name):

.NET API

```
...
public void Signup()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.SignUp("UserName@example.com", "Password",
```

```
        "FirstName", "LastName");
    }

    public async void SignupAsync()
    {
        var connection = new
            Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
        await connection.Accounts.Users.SignUpAsync("UserName@example.com",
            "Password", "FirstName", "LastName");
    }
    ...

```

You must to use an email address as the username.

1.1.1.2 Logging In

Log in registered user the class method **Login()** (**LoginAsync()**):

.NET API

```
...
public void LogIn()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");
}
...

```

Or asynchronous method:

.NET API

```
...
public async void LogInAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");
}
...

```

1.1.1.3 Getting User Info

In order to obtain information about the user, use the methods **GetByName()** or **GetByKey()** that return a **StiUser** object. This object contains full information about

the user (key of user role, workspace and root folder, OAuth identifier and type of authorization method, first name, last name, user name (email), avatar image, some flags (enabled, activated) and time when this user was created, last modified and last logged). Before calling the method **GetByName()** or **GetByKey()** you must log in as a user whose rights are allowed to have access to the necessary information.

.NET API

```
...
public void GetUserInfo()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var userJohn = connection.Accounts.Users.GetByName("John@example.com");
    var johnLastName = userJohn.LastName;
    var johnLogin = userJohn.LastLogin;

    var userScott = connection.Accounts.Users.GetByKey("ScottKey");
    var scottLastName = userScott.LastName;
    var scottLogin = userScott.LastLogin;

    var currentUser = connection.Accounts.Users.Current;
    var currentName = currentUser.UserName;
}
...
```

Or asynchronous method:

.NET API

```
...
public async void GetUserInfoAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");

    var userJohn = await
        connection.Accounts.Users.GetByNameAsync("John@example.com");
    var johnLastName = userJohn.LastName;
    var johnLogin = userJohn.LastLogin;

    var userScott = await
        connection.Accounts.Users.GetByKeyAsync("ScottKey");
}
```

```
var scottLastName = userScott.LastName;
var scottLogin = userScott.LastLogin;

var currentUser = connection.Accounts.Users.Current;
var currentName = currentUser.UserName;
}
...
```

1.1.1.4 Getting List of Users

To find or process information about users of the system, there is a method that allows you to get a list of all objects **StiUser**, to which the current user can access. Use the method **FetchAll()** (**FetchAllAsync()**).

.NET API

```
...
public void ProcessUsersInfo()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");
    var users = connection.Accounts.Users.FetchAll();

    //find user with last name "Smith"
    var mrSmith = users.First(a => a.FirstName == "Smith");
}
...
```

Asynchronous method:

.NET API

```
...
public async void ProcessUsersInfoAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");
    var users = await connection.Accounts.Users.FetchAllAsync();

    //is exist user with name "John"
    var isJohnExists = users.Any(a => a.LastName == "John");
}
...
```

1.1.1.5 Changing User Info

Passwords are stored in the system in the form of hashes and the `StiUser` object does not return hem. The method **`ChangePassword()`** (**`ChangePasswordAsync()`**) is used to change the password. Using parameters you must specify the current password and a new one:

.NET API

```
...
public void ChangeUserLastName()
{
    var connection = new
    Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var userJohn = connection.Accounts.Users.GetByName("John@example.com");
    userJohn.LastName = "Smith";
    userJohn.Save();
}
...
```

Asynchronous method:

.NET API

```
...
public async void ChangeUserLastNameAsync()
{
    var connection = new
    Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.Login("UserName@example.com",
    "Password");

    var userJohn = await
    connection.Accounts.Users.GetByName("John@example.com");
    userJohn.LastName = "Smith";
    await userJohn.Save();
}
...
```

1.1.1.6 Changing User Password

Passwords are stored in the system in the form of hashes and object **`StiUser`** not return hem. To change the password used method **`ChangePassword()`** (**`ChangePasswordAsync()`**). The parameters you must specify the current password and new:

.NET API

```
...
public void ChangeUserPassword()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var userJohn = connection.Accounts.Users.GetByName("John@example.com");
    userJohn.ChangePassword("JohnPassword", "NewPassword");
}
...
```

Asynchronous method:

.NET API

```
...
public async void ChangeUserPasswordAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");

    var userJohn = await
        connection.Accounts.Users.GetByNameAsync("John@example.com");
    await userJohn.ChangePasswordAsync("JohnPassword", "NewPassword");
}
...
```

1.1.1.7 Creating New User

Creating new users goes through the creation of a new object **StiUser** and it is storing through the methods **Save()** or **SaveAsync()**:

.NET API

```
...
public void CreateNewUser()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var newUser = connection.Accounts.Users.New();
    newUser.UserName = "UserName@example.com";
    newUser.Password = "UserPassword";
    newUser.Save();
}
```

```
...
```

Asynchronous method:

.NET API

```
...
public async void CreateNewUserAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");

    var newUser = connection.Accounts.Users.New();
    newUser.UserName = "UserName@example.com";
    newUser.Password = "UserPassword";
    await newUser.SaveAsync();
}
...
```

In addition you can create a user through **Roles** using the method **NewUser()**:

.NET API

```
...
public void NewUserFromRole()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var managerRole = connection.Accounts.Roles.ManagerRole;
    var newUser = managerRole.NewUser("NewUserName@example.com",
        "Password");

    connection.Accounts.Users.Logout();
}
...
```

1.1.1.8 Deleting User

Remove element by calling **Delete()** (**DeleteAsync()**) the object class **StiUser** or by calling one of the methods **DeleteByKey()**, **DeleteByKeyAsync()**, **DeleteByName()**, **DeleteByNameAsync()** from collection **StiServerConnection.Accounts.Users**:

.NET API

```
...
public void DeleteItem()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var user = connection.Accounts.Users.GetByKey("UserKey");
    if (user != null)
    {
        user.Delete();
    }

    connection.Accounts.Users.Logout();
}
...
```

Asynchronous example:

.NET API

```
...
public async void DeleteItemAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");

    await
        connection.Accounts.Users.DeleteByNameAsync("JohnSmith@example.com");

    await connection.Accounts.Users.LogoutAsync();
}
...
```

1.1.1.9 Logging Out

After all actions the user must log out. To do this, you can use the class methods **Logout()** or **LogoutAsync()**.

.NET API

```
...
public void Logout()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");
    connection.Accounts.Users.Logout();
}
```

```
...
```

Asynchronous method:

.NET API

```
...
public async void LogoutAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");
    await connection.Accounts.Users.LogoutAsync();
}
...
```

1.1.2 Roles

Stimulsoft Server supports a Role-based Access Control. Role is an object that determines the level of user access to system resources. Members having the same role have the same rights of access to system objects. The most important property of the role is **StiRole.Permissions**. It is an object of the **StiRolePermissions** class that describes a set of permissions (**StiPermissions**) for system objects:

Permission	Description
ItemCalendars	Permissions for Calendars
ItemCloudStorages	Permissions for Cloud Storages
ItemContactLists	Permissions for Contactlists
ItemDashboards	Permissions for Dashboards
ItemDataSources	Permissions for Datasources
ItemFiles	Permissions for Files
ItemFolders	Permissions for Folder
ItemReportSnapshots	Permissions for ReportSnapshots
ItemReportTemplates	Permissions for ReportTemplates
ItemSchedulers	Permissions for Schedulers

- › **None** - Denies all;
- › **Create** - Allows creating an item;
- › **Delete** - Allows deleting an item;
- › **Modify** - Allows modifying an item;
- › **Run** - Allows running an item;
- › **View** - Allows viewing an item;
- › **ModifyView** - Allows modifying and viewing an item;
- › **CreateDeleteModifyView** - Allows creating, deleting, modifying and viewing an item;
- › **RunView** - Allows running and viewing an item;
- › **All** - Allows any action with an item.

In order to use the **StiUserConnection** class you should create an instance of **StiServerConnection** and call one of its methods (**StiServerConnection.Accounts.Roles**).

The **StiRole** class provides access to information about the roles.

The role name or the key can obtain an instance of this class. For this use methods **GetByName()** (**GetByNameAsync()**) and **GetByKey()** (**GetByKeyAsync()**) respectively.

There is also the fastest way to get an instance of the current role by calling the **Current** property in the **StiServerConnection.Roles** class.

1.1.2.1 Creating and Saving Roles

To create a new role, you need to log in with a user name that has permission to work with roles, and create an object of the type **StiRole**, and then call its method **StiRole.Save()** (**StiRole.SaveAsync()**):

.NET API

```
...
public void CreateNewRole()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var role = connection.Accounts.Roles.New("UserRole");
    role.Permissions = connection.Accounts.Roles.ManagerRole.Permissions;
    role.Permissions.SystemMonitoring = StiPermissions.RunView;
    role.Save();

    connection.Accounts.Users.Logout();
}
```

```
...
```

Asynchronous example:

.NET API

```
...
public async void CreateNewRoleAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");

    var role = connection.Accounts.Roles.New("CustomRole");
    role.Permissions.SystemUpdate = StiPermissions.All;
    role.Permissions.ItemSchedulers = StiPermissions.CreateDeleteModifyView;
    await role.SaveAsync();

    await connection.Accounts.Users.LogoutAsync();
}
...
```

1.1.2.2 Getting Role Info

In order to obtain information about the role, use the methods **GetByKey()** or **GetByKeyAsync()** that return an object **StiRole**. However, a more convenient way, without requiring the key is the use of one of the objects **StiServerConnection.Accounts.Roles.AdministratorRole**, **StiServerConnection.Accounts.Roles.ManagerRole**, **StiServerConnection.Accounts.Roles.UserRole**. Before calling this methods you must log in as a user whose rights are allowed to have access to the necessary information.

.NET API

```
...
public void GetRoleInfo()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var userRole = connection.Accounts.Roles.GetByKey("RoleKey");
    var roleDescription = userRole.Description;

    var currentRole = connection.Accounts.Roles.Current;
    var currentRoleName = currentRole.Name;
}
```

```
connection.Accounts.Users.Logout();
}
...
```

Asynchronous method:

.NET API

```
...
public async void GetRoleInfoAsync()
{
    var connection = new
    Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
    "Password");

    var managerRole = connection.Accounts.Roles.ManagerRole;
    var managerCreateDate = managerRole.Created;
    var managerPermissions = managerRole.Permissions;

    var currentRole = connection.Accounts.Roles.Current;
    var currentRoleName = currentRole.Name;

    await connection.Accounts.Users.LogoutAsync();
}
...
```

1.1.2.3 Getting List of Roles

To find or process information about roles of the system, there is a method that allows you to get a list of all objects **StiRole**, to which the current user can access. Use the method **FetchAll()** (**FetchAllAsync()**).

.NET API

```
...
public void ProcessUsersInfo()
{
    var connection = new
    Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var roles = connection.Accounts.Roles.FetchAll();

    //find role with full access to system monitoring
    var monitoringAdministrator = roles.First(a =>
    a.Permissions.SystemMonitoring == StiPermissions.All);

    connection.Accounts.Users.Logout();
}
```

```
...
```

Asynchronous example:

.NET API

```
...
public async void ProcessUserInfoAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");

    var roles = await connection.Accounts.Roles.FetchAllAsync();

    //is exist role with name "ReportAdministrator"
    var isReportAdministrator = roles.Any(a => a.Name ==
        "ReportAdministrator");

    await connection.Accounts.Users.LogoutAsync();
}
...
```

1.1.2.4 Deleting Role

Remove the role by calling **DeleteByKey()** (**DeleteByKeyAsync()**). Second way is creating an object of the type **StiRole**, and then calling its method **StiRole.Delete()** (**StiRole.DeleteAsync()**):

.NET API

```
...
public void DeleteRole()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    connection.Accounts.Roles.DeleteByKey("RoleKey");

    connection.Accounts.Users.Logout();
}
...
```

Asynchronous method:

.NET API

```
...
public async void DeleteRoleAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");

    var managerRole = connection.Accounts.Roles.ManagerRole;

    await managerRole.DeleteAsync();

    await connection.Accounts.Users.LogoutAsync();
}
...
```

1.1.3 Workspaces

For separation of data from different working groups Stimulsoft Server uses the concept of workspaces. Some users work in one workspace and their data are available to other users. Users working in different workspaces do not have access to each other.

In order to use the **StiWorkspaceConnection** class you should create an instance of **StiServerConnection** and call one of its methods (**StiServerConnection.Accounts.Workspaces**).

The **StiWorkspace** class provides access to information about the roles. The workspace key can obtain an instance of this class. Use for this methods **GetByKey()** (**GetByKeyAsync()**).

There is also the fastest way to get an instance of the current workspace by calling the **Current** property in the **StiServerConnection.Accounts.Workspaces** class.

1.1.3.1 Creating and Saving

To create a new workspace, you need to log in with a user name that has permission to work with workspace, and create an object of the type **StiWorkspace**, and then call its method **StiWorkspace.Save()** (**StiWorkspace.SaveAsync()**):

.NET API

```
...
public void CreateNewWorkspace()
{
    var connection = new
```

```
Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
connection.Accounts.Users.Login("UserName@example.com", "Password");

var workspace = connection.Accounts.Workspaces.New("Company");
var workspaceKey = workspace.Key;
workspace.Save();

connection.Accounts.Users.Logout();
}
...
```

An asynchronous example:

.NET API

```
...
public async void CreateNewWorkspaceAsync()
{
    var connection = new
    Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
    "Password");

    var workspace = connection.Accounts.Workspaces.New("Company");
    var workspaceKey = workspace.Key;
    await workspace.SaveAsync();

    await connection.Accounts.Users.LogoutAsync();
}
...
```

1.1.3.2 Getting Workspace Info

In order to obtain information about the workspace, use the methods **GetByKey()** or **GetByKeyAsync()** that return the **StiWorkspace** object. Before calling these methods you must log in as a user whose rights are allowed to have access to the necessary information.

.NET API

```
...
public void GetWorkspaceInfo()
{
    var connection = new
    Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var workspace = connection.Accounts.Workspaces.GetByKey("WorkspaceKey");
    var workspaceCompany = workspace.Company;
}
```

```
var currentWorkspace = connection.Accounts.Workspaces.Current;
var currentWorkspaceKey = currentWorkspace.Key;

connection.Accounts.Users.Logout();
}
...
```

An asynchronous method:

.NET API

```
...
public async void GetWorkspaceInfoAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");

    var workspace = connection.Accounts.Workspaces.GetByKey("WorkspaceKey");
    var workspaceCompany = workspace.Company;

    var currentWorkspace = connection.Accounts.Workspaces.Current;
    var currentWorkspaceKey = currentWorkspace.Key;

    await connection.Accounts.Users.LogoutAsync();
}
...
```

1.1.3.3 Getting List of Workspaces

To find or process information about workspaces of the system, there is a method that allows you to get a list of all objects **StiWorkspace**, to which the current user can access. Use the method **FetchAll()** (**FetchAllAsync()**).

.NET API

```
...
public void ProcessWorkspacesInfo()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var workspace = connection.Accounts.Workspaces.FetchAll();

    //find workspace of the company with the name Northwind
    var northwindWorkspace = workspace.First(a => a.Company == "Northwind");

    connection.Accounts.Users.Logout();
}
```

```
...
```

An asynchronous example:

.NET API

```
...
public async void ProcessWorkspacesInfoAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");

    var workspace = connection.Accounts.Workspaces.FetchAll();

    //find workspaces created to 01.01.2015
    var newWorkspaces = workspace.First(a => a.Created <= new DateTime(2015,
        01, 01));

    await connection.Accounts.Users.LogoutAsync();
}
...
```

1.1.3.4 Deleting Workspace

To delete a workspace, you must have supervisor rights. A workspace in which the supervisor is registered cannot be deleted. Remove the workspace by calling **DeleteByKey()** (**DeleteByKeyAsync()**). Second way is creating an object of the **StiWorkspace** type, and then calling its method **StiWorkspace.Delete()** (**StiWorkspace.DeleteAsync()**):

.NET API

```
...
public void DeleteWorkspace()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    connection.Accounts.Workspaces.DeleteByKey("WorkspaceKey");

    connection.Accounts.Users.Logout();
}
...
```

An asynchronous method:

.NET API

```
...
public async void DeleteWorkspaceAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");

    var workspace =
        connection.Accounts.Workspaces.GetByKeyAsync("WorkspaceKey");

    await workspace.Result.DeleteAsync();

    await connection.Accounts.Users.LogoutAsync();
}
...
```

1.1.4 Items

The basis of the **Stimulsoft Server** functionality is operations on items. The item is an object that can be visually observed in the object tree on the left side of the interface of the client application. All items are inherited from the root abstract class **StiItem** and, there is one of the following classes depending on the functionality provided:

- **StiCalendarItem** - used to create a scheduler;
- **StiContactListItem** - used for sending reports via e-mail;
- **StiDataSourceItem** - used to connect external data sources;
- **StiFileItem** - used to connect external data sources;
- **StiFolderItem** - provides a hierarchical tree structure elements;
- **StiReportSnapshotItem** - rendered data report;
- **StiReportTemplateItem** - report template for building;
- **StiSchedulerItem** - used to automate actions.

1.1.4.1 Creating and Saving Items

Position in the hierarchical tree structure of elements defined by the connection element and folders. Therefore, to create the item, you must first create an object of type **StiFolderItem**, specifying its position in the tree, and then use one of the methods to create an item of a particular type. The root element of the tree is represented by an instance of the class **StiFolderItem**:

StiServerConnection.Items.Root. After defining properties of the new item is necessary to perform his method **StiItem.Save()** or **StiItem.SaveAsync()**. The following example shows how to create a folder in the root of the tree of elements and add the calendar (describes Monday) to this folder:

.NET API

```
...
public void CreateNewCalendarItem()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var folderItem = connection.Items.Root.NewFolder("folder");
    folderItem.Save();

    var calendarItem = folderItem.NewCalendar("NewCalendar");
    calendarItem.Dates.Add(new StiCalendarDate("Monday",
        StiDaysOfWeek.Monday));
    calendarItem.Save();
}
...
```

Asynchronous method:

.NET API

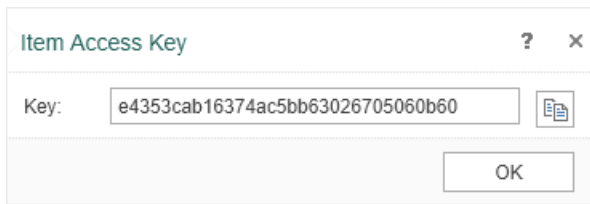
```
...
public async void CreateNewCalendarItemAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");

    var folderItem = connection.Items.Root.NewFolder("folder");
    await folderItem.SaveAsync();

    var calendarItem = folderItem.NewCalendar("NewCalendar");
    calendarItem.Dates.Add(new StiCalendarDate("Monday",
        StiDaysOfWeek.Monday));
    await calendarItem.SaveAsync();
}
...
```

1.1.4.2 Sharing Items

To work with elements uses a unique identifier - keys. They are assigned automatically when you create elements.

A dialog box titled "Item Access Key" with a question mark icon and a close button (X). It contains a text input field labeled "Key:" with the value "e4353cab16374ac5bb63026705060b60". To the right of the input field is a copy icon. At the bottom right is an "OK" button.

Item Access Key	
Key:	e4353cab16374ac5bb63026705060b60
OK	

Press the **Access Key** command from the **More** menu to get the key.

The key can be used to specify a particular item in the API and to access an item outside over HTTP. This example demonstrates a simple HTML-page that provides access to the report in the public domain:

.NET API

```
...
<html>
<head>
  <title>Sharing example</title>
</head>
<body>
  <p style="font-size: 40px;">The example of shared report.</p>
  <br>
  <iframe src="http://localhost:40010/
    share/13f5d51dd5294a9483facdf61299000a"></iframe>
</body>
</html>
...
```

1.1.4.3 Getting Item

To get an existing item is necessary to know the key that is passed to the method as a parameter. Use methods **StiItem.GetByKey()** and **StiItem.GetByKeyAsync()**:

.NET API

```
...
public void GetCalendarItem()
{
    var connection = new
    Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var item = connection.Items.GetByKey("CalendarItemKey");
    if (item != null)
    {
        var calendarItem = item as StiCalendarItem;
        if (calendarItem != null)
    }
}
```

```
    {  
        var calendarItemDescription = calendarItem.Description;  
    }  
}  
}  
...
```

Asynchronous method:

.NET API

```
...  
public async void GetCalendarItemAsync()  
{  
    var connection = new  
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");  
    await connection.Accounts.Users.LoginAsync("UserName@example.com",  
        "Password");  
  
    var item = await connection.Items.GetByKeyAsync("CalendarItemKey");  
    if (item != null)  
    {  
        var calendarItem = item as StiCalendarItem;  
        if (calendarItem != null)  
        {  
            var calendarItemDescription = calendarItem.Description;  
        }  
    }  
}  
...
```

1.1.4.4 Getting List of Items

To find or process items, there is a method that allows you to get a list of all objects **StiItem**, to which the current user can access. Use the method **FetchAll()** (**FetchAllAsync()**).

.NET API

```
...  
public void ProcessItems()  
{  
    var connection = new  
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");  
    connection.Accounts.Users.Login("UserName@example.com", "Password");  
    var items = connection.Items.Root.FetchChilds();  
  
    //find folder with name "Folder1"  
    var folder1 = items.First(a => a.Name == "Folder1");  
}  
...
```

Asynchronous method:

.NET API

```
...
public async void ProcessItemsAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");
    var items = await connection.Items.Root.FetchChildsAsync();

    //is exist any folder
    var isFolder = items.Any(a => a.IsFolder);
}
...
```

1.1.4.5 Deleting Item

Remove element by calling **Delete()** (**DeleteAsync()**):

.NET API

```
...
public void DeleteItem()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var item = connection.Items.GetByKey("CalendarItemKey");
    if (item != null)
    {
        //Delete item with skipping undeletable items
        item.Delete(true, true);
    }
}
...
```

Asynchronous method:

.NET API

```
...
public async void DeleteItemAsync()
{
    var connection = new
```

```
Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
await connection.Accounts.Users.LoginAsync("UserName@example.com",
"Password");

var item = await connection.Items.GetByKeyAsync("CalendarItemKey");
if (item != null)
{
    //Delete item without moving it into the recycle bin
    await item.DeleteAsync(false);
}
...

```

1.1.4.6 Resource Item

Some element types (**StiFileItem**, **StiReportSnapshotItem**, **StiReportTemplateItem**) include resources. Using the methods **UploadFromFile()**, **UploadFromFileAsync()**, **UploadFromArray()**, **UploadFromArrayAsync()**, **DownloadToFile()**, **DownloadToFileAsync()**, **DownloadToArray()**, **DownloadToArrayAsync()**. You can manipulate the data contained in these resources. For example, creating a file on the server with the loading data into it looks like this:

.NET API

```
...
public void CreateNewFile()
{
    var connection = new
    Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var content = File.ReadAllBytes(@"C:\testfile.xml");

    var newFile = connection.Items.Root.NewFile("TestFile.xml");
    newFile.Save();

    newFile.UploadFromArray(content);
}
...

```

Asynchronous method:

.NET API

```
...
public async void CreateNewTemplateAsync()
{
    var connection = new

```

```
Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
await connection.Accounts.Users.LoginAsync("UserName@example.com",
"Password");

var newTemplate = connection.Items.Root.NewReportTemplate("Master-
Detail");
await newTemplate.SaveAsync();

await newTemplate.UploadFromFileAsync(@"C:\master-detail.mrt");
}
...
```

Loading item from the server and saving it to a file on the local computer as follows:

.NET API

```
...
public void DownloadFile()
{
    var connection = new
    Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var newFile = connection.Items.GetByKey("FileItemKey");
    if (newFile is StiFileItem)
    {
        var data = (newFile as StiFileItem).DownloadToArray();
        File.WriteAllBytes(@"c:\newfile", data);
    }
}
...
```

Asynchronous method:

.NET API

```
...
public async void DownloadFileAsync()
{
    var connection = new
    Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
"Password");

    var report = await
    connection.Items.GetByKeyAsync("ReportTemplateItemKey");
    if (report is StiReportTemplateItem)
    {
        await (report as StiReportTemplateItem).DownloadToFileAsync(@"C:
\Master-Detail.mrt");
    }
}
```

```
}  
...
```

1.1.4.7 Attaching Items to Reports and Files

The elements of type **StiFileItem** and **StiReportTemplateItem** can attach resource elements (**StiFileItem**, **StiReportSnapshotItem**, **StiReportTemplateItem**). This is done using the method **AttachItem(StiResourceItem resourceItem)**. Detaching the elements is performed using the method **DetachItem(StiResourceItem resourceItem)**. This example shows a file attachment into a report template:

.NET API

```
...  
public void AttachItemToReportTemplate()  
{  
    var connection = new  
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");  
    connection.Accounts.Users.Login("UserName@example.com", "Password");  
  
    // Report Template  
    var reportTemplateItem =  
        connection.Items.Root.NewReportTemplate("report-template");  
    reportTemplateItem.Save();  
    reportTemplateItem.UploadFromFile(@"C:\report.mrt");  
  
    // Attached File  
    var attachedFile = connection.Items.Root.NewFile("attach",  
        StiFileType.Image);  
    attachedFile.Save();  
    attachedFile.UploadFromFile(@"C:\image.png");  
  
    // Attach file to report template  
    reportTemplateItem.AttachItem(attachedFile);  
  
    // Detach file from report template  
    reportTemplateItem.DetachItem(attachedFile);  
}  
...
```

Asynchronous method for a XSD-file attachment into a XML-file:

.NET API

```
...  
public async void AttachXsdToXmlAsync()  
{  
    var connection = new  
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");  
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
```

```

        "Password");

    // Xml File
    var xmlFile = connection.Items.Root.NewFile("data.xml",
        StiFileType.Xml);
    await xmlFile.SaveAsync();
    await xmlFile.UploadFromFileAsync(@"C:\data.xml");

    // Xsd File
    var xsdFile = connection.Items.Root.NewFile("data.xsd",
        StiFileType.Xsd);
    await xsdFile.SaveAsync();
    await xsdFile.UploadFromFileAsync(@"C:\data.xsd");

    // Attach Xsd to Xml
    await xmlFile.AttachItem(xsdFile);

    // Detach item
    await xmlFile.DetachItem(xsdFile);
}
...

```

1.1.4.8 Running Report

The result of creating Report Template in the tree is a new element type **StiReportTemplateItem**. This object describes the report without data. Designing items of the report is made through the Navigator interface, but managing the construction is possible by means of appropriate methods **StiReportTemplateItem.Run()** and **StiReportTemplateItem.RunAsync()**. The parameter specifies the item **StiReportTemplateItem** or **StiFileItem** where you saved the rendered report. The object must be created before saving. The item **StiFileItem** can be one of the possible types of enumeration **StiFileType** (ReportSnapshot, PDF, XPS, PowerPoint, HTML, Text, RichText, Word, OpenDocumentWriter, Excel, OpenDocumentCalc, Data, Image, XML, XSD, CSV, DBF). One of these values are specified in the method **StiServerConnection.Items.Root.NewFile()** as a parameter. The following example creates a report template, loads data into it and runs the report, after which the result is stored in the report snapshot:

.NET API

```

...
public void RunReportToShapshot()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var reportTemplateItem =

```

```
connection.Items.Root.NewReportTemplate("report-template");
reportTemplateItem.Save();
reportTemplateItem.UploadFromFile(@"C:\report.mrt");

var reportSnapshotItem =
connection.Items.Root.NewReportSnapshot("report-snapshot");
reportSnapshotItem.Save();
reportTemplateItem.Run(reportSnapshotItem);
}
...
```

Asynchronous method creates a report template, converts data and runs the report, after which the result is stored in the PDF file:

.NET API

```
...
public async void RunReportToPdfFileAsync()
{
    var connection = new
    Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
    "Password");

    var reportTemplateItem =
    connection.Items.Root.NewReportTemplate("report-template");
    await reportTemplateItem.SaveAsync();
    await reportTemplateItem.UploadFromFileAsync(@"C:\report.mrt");

    var pdfReportItem = connection.Items.Root.NewFile("report.pdf",
    StiFileType.Pdf);
    await pdfReportItem.SaveAsync();
    await reportTemplateItem.RunAsync(pdfReportItem);
}
...
```

1.1.4.9 Exporting Report Snapshot

The result of the construction of the report from **StiReportTemplateItem** is **StiReportSnapshotItem**. Export data from it can be done by methods **StiReportSnapshotItem.Export()** and **StiReportSnapshotItem.ExportAsync()**. The data is stored in the item **StiFileItem**. It should be created in advance with a specific type the constructor, as described in the previous chapter. Optional parameter method **StiReportSnapshotItem.Export()** is an object **StiTextExportSet**, which describes many options of visualization in the report to export. This example demonstrates exporting the report snapshot to the Excel Document:

.NET API

```
...
public void ExportSnapshotToExcel()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var reportSnapshotItem =
        connection.Items.Root.NewReportSnapshot("report-snapshot");
    reportSnapshotItem.Save();
    reportSnapshotItem.UploadFromFile(@"C:\report-snapshot.mdc");

    var fileItemExcel = connection.Items.Root.NewFile("ExcelDocument",
        StiFileType.Excel);
    fileItemExcel.Save();

    reportSnapshotItem.Export(fileItemExcel,
        new StiTextExportSet
        {
            BorderType = Report.Export.StiTxtBorderType.UnicodeDouble,
            CutLongLines = true,
            KillSpaceGraphLines = true
        });
}
...
```

An example of an asynchronous method. Exporting a report snapshot to the XML file:

.NET API

```
...
public async void ExportSnapshotToXmlAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");

    var reportSnapshotItem =
        connection.Items.Root.NewReportSnapshot("report-snapshot");
    await reportSnapshotItem.SaveAsync();
    await reportSnapshotItem.UploadFromFileAsync(@"C:\report-snapshot.mdc");

    var fileItemXml = connection.Items.Root.NewFile("XmlFile",
        StiFileType.Xml);
    await fileItemXml.SaveAsync();

    await reportSnapshotItem.ExportAsync(fileItemXml);
}
...
```

1.1.4.10 Scheduling Actions

Stimulsoft Server supports flexible system schedulers that allow the automation of various actions and events schedule. The object **StiSchedulerItem** inherited from **StiItem** is used to work with the scheduler. After creating an instance of this object, you must add actions using the method **StiSchedulerItem.AddRunReportAction()**. The parameters of this overloaded method take several sets of objects that define the functionality of the scheduler. The minimum set of parameters is as follows:

➤ **AddRunReportAction**

(**StiReportTemplateItem** reportTemplateItem, **StiReportSnapshotItem** reportSnapshotItem) – at work scheduler builds a report template **reportTemplateItem** and stores it to a snapshot **reportSnapshotItem**;

➤ **AddRunReportAction**

(**StiReportTemplateItem** reportTemplateItem, **StiFileItem** fileItem) – at work scheduler builds a report template **reportTemplateItem** and export it to a file **fileItem**.

➤ **AddRunReportAction**

(**StiReportTemplateItem** reportTemplateItem, **StiContactListItem** contactListItem, **StiFileType** fileType) – at work scheduler builds a report template **reportTemplateItem**, export it to a file of **fileType** type and sends the results by e-mail to contacts from **contactListItem**.

To start the scheduler immediately, the **StiSchedulerItem.Run()** (**StiSchedulerItem.RunAsync()**) method is used. This example creates a scheduler to render the report as a snapshot:

.NET API

```
...
public void CreateSchedulerAndRunReportToSnapshot ()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    // Create folder
    var folderItem = connection.Items.Root.NewFolder("folder");
    folderItem.Save();

    // Create report template
    var reportTemplateItem = folderItem.NewReportTemplate("report-
        template");
    reportTemplateItem.Save();
    reportTemplateItem.UploadFromFile(@"c:\ReportTemplate.mrt");
}
```

```
// Create report snapshot
var reportSnapshotItem = folderItem.NewReportSnapshot("report-
snapshot");
reportSnapshotItem.Save();

// Create scheduler
var schedulerItem = folderItem.NewScheduler("scheduler",
StiSchedulerIdent.Once);
schedulerItem.AddRunReportAction(reportTemplateItem,
reportSnapshotItem);
schedulerItem.Save();

// Run scheduler
schedulerItem.Run();

connection.Accounts.Users.Logout();
}
...
```

This asynchronous method creates a scheduler which renders the template to the PDF file:

.NET API

```
...
public async void CreateSchedulerAndRunReportToFileAsync()
{
    var connection = new
    Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
    "Password");

    // Create folder
    var folderItem = connection.Items.Root.NewFolder("folder");
    await folderItem.SaveAsync();

    // Create report template
    var reportTemplateItem = folderItem.NewReportTemplate("report-
template");
    await reportTemplateItem.SaveAsync();
    await reportTemplateItem.UploadFromFileAsync(@"c:\ReportTemplate.mrt");

    // Create file
    var fileItem = folderItem.NewFile("file", StiFileType.Pdf);
    await fileItem.SaveAsync();

    // Create scheduler
    var schedulerItem = folderItem.NewScheduler("scheduler",
    StiSchedulerIdent.Once);
    schedulerItem.AddRunReportAction(reportTemplateItem, fileItem);
    await schedulerItem.SaveAsync();

    // Run scheduler
    await schedulerItem.RunAsync();
}
```

```
    await connection.Accounts.Users.LogoutAsync();  
}  
...
```

This example creates a scheduler which renders the template to the PDF file and sends it to the email addresses listed in the contact element:

.NET API

```
...  
public async void CreateSchedulerAndRunReportToContactsAsync()  
{  
    var connection = new  
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");  
    await connection.Accounts.Users.LoginAsync("UserName@example.com",  
        "Password");  
  
    // Create folder  
    var folderItem = connection.Items.Root.NewFolder("folder");  
    await folderItem.SaveAsync();  
  
    // Create report template  
    var reportTemplateItem = folderItem.NewReportTemplate("report-  
template");  
    await reportTemplateItem.SaveAsync();  
    await reportTemplateItem.UploadFromFileAsync(@"c:\ReportTemplate.mrt");  
  
    // Create contact list  
    var contactListItem = folderItem.NewContactList("contacts",  
        "smith@example.com, jones@example.com");  
    contactListItem.Save();  
  
    // Create scheduler  
    var schedulerItem = folderItem.NewScheduler("scheduler",  
        StiSchedulerIdent.Once);  
    schedulerItem.AddRunReportAction(reportTemplateItem, contactListItem,  
        StiFileType.Pdf);  
    await schedulerItem.SaveAsync();  
  
    // Run scheduler  
    await schedulerItem.RunAsync();  
  
    await connection.Accounts.Users.LogoutAsync();  
}  
...
```

The scheduler can have two states (describes the objects **StiSchedulerStatus**) – running (track events and time, and depending on this, the scheduler can run) and stopped (not tracked any external state). The state of the scheduler is described by

two values – Started or Stopped. To set the scheduler status the **StiSchedulerItem.SetStatus()** (**StiSchedulerItem.SetStatusAsync()**) method is used. This example gets the status of the scheduler:

.NET API

```
...
public void CreateSchedulerAndGetState()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var schedulerItem = connection.Items.Root.NewScheduler("scheduler",
        StiSchedulerIdent.Hourly).Save();

    var status = schedulerItem.GetStatus();

    Debug.WriteLine(status == StiSchedulerStatus.Stopped ? "Scheduler
        Stopped" : "Scheduler Started");

    connection.Accounts.Users.Logout();
}
...
```

This example sets a scheduler status asynchronously:

.NET API

```
...
public async void CreateSchedulerAndSetStateAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");

    var schedulerItem = connection.Items.Root.NewScheduler("scheduler",
        StiSchedulerIdent.Hourly).SaveAsync();

    await schedulerItem.Result.SetStatusAsync(StiSchedulerStatus.Stopped);

    await connection.Accounts.Users.LogoutAsync();
}
...
```

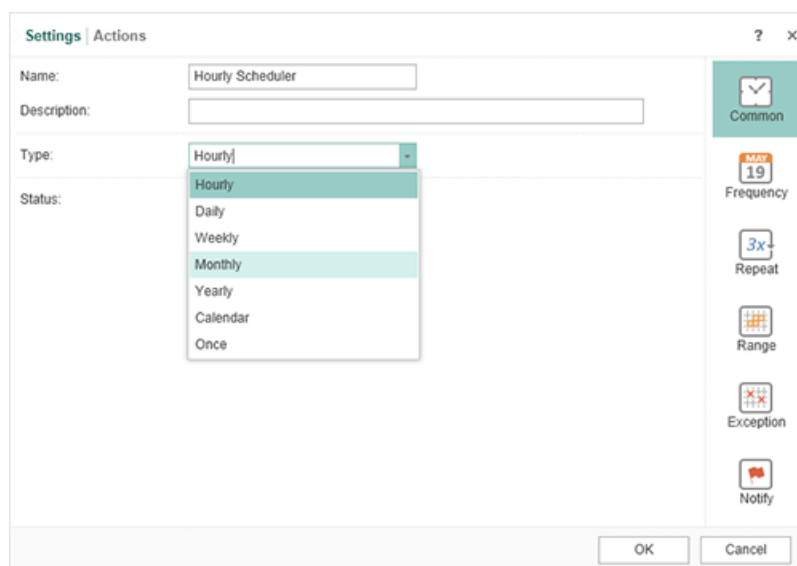
1.1.4.11 Scheduling Frequency

The flexibility of the scheduler is implemented through the variety of options for setting the frequency of execution of actions. There are seven types of the scheduler

implemented in the classes:

- StiCalendarScheduler
- StiDailyScheduler
- StiHourlyScheduler
- StiMonthlyScheduler
- StiOnceScheduler
- StiWeeklyScheduler
- StiYearlyScheduler

In the **Stimulsoft Server** interface the type of a scheduler is set as follows:



There are special methods that return a new class of type **StiSchedulerItem** to make it easier to work with schedulers of different types:

Name	Description
SetHourlyFrequency (int runAtMinute)	Hourly launch actions in runAtMinute minutes;
SetDailyFrequency (int runAtHour, int runAtMinute = 0)	Daily launch actions in runAtHour hours runAtMinute minutes;
SetWeeklyFrequency (StiDaysOfWeek daysOfWeek, int runAtHour = 0, int runAtMinute = 0)	Weekly launch actions in daysOfWeek day of week, runAtHour hours, runAtMinute minutes. StiDaysOfWeek it is the enumeration of days of the

	week, empty value (None) and value for all days of the week (All);
SetMonthlyFrequency (StiMonths runAtMonth, StiDays runAtDay, int runAtHour = 0, int runAtMinute = 0)	Monthly launch actions in runAtMonth month, runAtDay day, runAtHour hours, runAtMinute minutes. This type of the scheduler starts actions in concrete day of month at the appointed time: 12th or last day of month. StiMonths it is the enumeration of months of the year, empty value (None) and value for all months of the year (All). StiDays it is the enumeration of days of the month, empty value (None), value for all days of the month (All) and value for the last day of the month (Last);
SetMonthlyFrequency (StiMonths runAtMonth, StiDaysOfWeek daysOfWeek, StiNumberOfDays numberOfDays, int runAtHour = 0, int runAtMinute = 0)	Monthly launch actions in runAtMonth month, daysOfWeek day of the week, numberOfDays number of week, runAtHour hours, runAtMinute minutes. This type of the scheduler starts actions in a relative day of a month at the appointed time: every second Thursday of month or last Friday of month. StiMonths is the enumeration of months of the year, empty value (None) and value for all months of the year (All). StiDaysOfWeek it is the enumeration of days of the week, empty value (None) and value for all days of the week (All). StiNumberOfDays it is the enumeration of values First, Second, Third, Fourth, Fifth, empty value (None) and value for all items (All);
SetYearlyFrequency (StiMonths runAtMonth, StiDays runAtDay, int runAtHour = 0, int runAtMinute = 0)	Yearly launch actions in runAtMonth month, runAtDay day, runAtHour hours, runAtMinute minutes. StiMonths it is the enumeration of months of the year,

	empty value (None) and value for all months of the year (All). StiDays it is the enumeration of days of the month, empty value (None), value for all days of the month (All) and value for the last day of the month (Last);
SetCalendarFrequency (StiCalendarItem calendarItem, int runAtHour = 0, int runAtMinute = 0)	launch actions in runAtHour hours, runAtMinute minutes by dates of a calendar calendarItem (the StiCalendarItem element, has to be created in advance);
SetExceptionCalendar (StiCalendarItem exceptionCalendarItem)	set a calendar of exceptions - a list of dates on which the action will not be executed;
SetDateRange (DateTime? startDate, DateTime? endDate)	set the range of dates of activity of the scheduler;
SetRunEvery (int runEvery)	sets the shift in the launch of actions the scheduler, for example, if set to 2 then the event will be run every second time of activation the scheduler, if set to 5 - every fifth time;
SetExcludeWeekendDays (bool excludeWeekendDays)	if set to True then excludes the execution of actions of the scheduler in the weekends;
SetTimeZone (string timeZone)	set required time zone;
SetRunOnce()	if set to True then actions of the scheduler are executed once;
SetRepeat (int repeatCount, float repeatRate, StiRepeatType repeatType)	repeat actions of the scheduler repeatCount times with the interval of repeatRate hours or minutes (StiRepeatType it is the enumeration of values Hours and Minutes).

This example sets start of the scheduler's actions each 8 hours:

.NET API

```
...
public void CreateSchedulerAndSetFrequencyTo8h()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    var schedulerItem = connection.Items.Root.NewScheduler("scheduler",
        StiSchedulerIdent.Hourly).Save();

    schedulerItem = schedulerItem.SetHourlyFrequency(8);
    schedulerItem.Save();

    connection.Accounts.Users.Logout();
}
...
```

An asynchronous method sets start of the scheduler actions by dates of a calendar:

.NET API

```
...
public async void CreateSchedulerAndSetFrequencyCalendarAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");

    var calendarItem = connection.Items.Root.NewCalendar("calendar");
    calendarItem.Dates.Add(new StiCalendarDate("First day of 2020", new
        DateTime(2020, 01, 01)));
    await calendarItem.SaveAsync();

    var schedulerItem = connection.Items.Root.NewScheduler("scheduler",
        StiSchedulerIdent.Hourly);

    schedulerItem = schedulerItem.SetCalendarFrequency(calendarItem);

    await schedulerItem.SaveAsync();

    await connection.Accounts.Users.LogoutAsync();
}
...
```

1.1.4.12 Data Sources

For working with data sources, Stimulsoft Server has a special type of a **StiItem** – **StiDataSourceItem**, which has some specific features and capabilities.

StiDataSourceItem is an element that provides a connection to a database and retrieves the necessary data for displaying in the report. There are several methods to connect a specific database:

- **SetFirebird** – connects to an existing DataSource adapter Firebird SQL;
- **SetMsSQL** – connects to an existing DataSource adapter MS SQL Server;
- **SetMySQL** – connects to an existing DataSource adapter MySQL Server;
- **SetODBC** – connects to an existing DataSource adapter Open Database Connectivity (ODBC);
- **SetOracle** – connects to an existing DataSource adapter Oracle Database;
- **SetPostgreSQL** – connects to an existing DataSource adapter Postgre SQL;
- **SetSQLCE** – connects to an existing DataSource adapter Microsoft SQL Server Compact;
- **SetSQLite** – connects to an existing DataSource adapter SQLite DB;

As an argument to any of these methods, you must specify the connection string in the format supported by the specified database. There is the **StiDataQueryItem** class for receiving data set from a database. It describes the SQL query for the selected database. Creating an instance of this class can be done by using the **StiDataSourceItem.NewQuery** method. It allows you to create an SQL query to retrieve a dataset from the database. The parameters of this method are the name of the query and the text of the query in a format supported by the selected database.

StiDataQueryItem.Run or **StiDataQueryItem.RunAsync** methods is used to run the SQL query. As optional parameters restrictions on the size of the final sample are used: index of the first element and the number of elements of the dataset to be retrieved. The execution method is an instance of the system DataTable class with a set of requested data.

This example shows how to create a data source and run a query to obtain the number of records in a table «Items»:

.NET API

```
...
public void RunDataSourceQueryCount ()
{
    var connection = new
    Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    const string connectionString =
    "Server=localhost;Database=testbase;Uid=root;Pwd=123;Connection
    Timeout=30;";
```

```

var dataSource = connection.Items.Root.NewDataSource("test");
dataSource.SetMySQL(connectionString);
dataSource.Save();

var dataQueryCount = dataSource.NewQuery("test query", "select count(*)
from items").Save();
var tableCount = dataQueryCount.Run();
var count = (long)tableCount.Rows[0].ItemArray[0];
}
...

```

This asynchronous method creates a `DataSource` and runs a query to retrieve the first 500 elements of the table «Items»:

.NET API

```

...
public async void RunDataSourceQuerySelectAsync()
{
    var connection = new
    Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
    "Password");

    const string connectionString =
    "Server=localhost;Database=testbase;Uid=root;Pwd=123;Connection
    Timeout=30;";
    var dataSource =
    connection.Items.Root.NewDataSource("test").SetMySQL(connectionString).S
    ave();

    var dataQuery = dataSource.NewQuery("test query", "select * from
    items").Save();
    var tableBeforeResult = await dataQuery.RunAsync(0, 500);
}
...

```

1.1.4.13 File

For working with uploaded files, Stimulsoft Server has a special type of a **StiItem** - **StiFileItem**, which has some specific features and capabilities. **StiFileItem** is an element that provides storing files, uploaded by users. Some files contain data that can be used in the reports as a data source – these are XLS documents, CSV tables, DBF databases, JSON strings etc. Getting the structured data from these files is possible using the **StiFileItem.GetData (StiFileItem.GetDataAsync)** method. At creation of a **StiFileItem** you must specify the type of data which the loaded file contains. Allowed values listed in enumeration **StiFileType**:

- Unknown - Unsupported file type

- ReportSnapshot - Rendered report
- Pdf – PDF file
- Xps – XPS file
- Html – HTML file
- Text – Text file
- RichText – RichText file format (RTF)
- Word – MS Word document file
- Excel – MS Excel document file
- PowerPoint – MS PowerPoint presentation file
- OpenDocumentWriter – OpenDocument file for Writer
- OpenDocumentCalc – OpenDocument file for Calc
- Data - One of multiple data format
- Image - One of multiple image format
- Xml – XML file
- Xsd – XSD file
- Csv – CSV file
- Dbf – DBF file
- Sylk – SYLK file
- Dif – DIF file
- Json - Data in the JSON format

These examples create an element of the **StiFileItem** type, load a file into it and receive the data as a table from it:

.NET API

```
...
public void FileItemGetData()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    connection.Accounts.Users.Login("UserName@example.com", "Password");

    const string excelFileName = @"C:\sample.xls";

    var fileItem = connection.Items.Root.NewFile("ExcelFile",
        StiFileType.Excel).Save();
    fileItem.UploadFromFile(excelFileName);

    var table1 = fileItem.GetData("Sheet1");
    var rowCount = table1.Rows.Count;

    connection.Accounts.Users.Logout();
}
...
```

An asynchronous method:

.NET API

```
...
public async void FileItemGetDataAsync()
{
    var connection = new
        Stimulsoft.Server.Connect.StiServerConnection("localhost:40010");
    await connection.Accounts.Users.LoginAsync("UserName@example.com",
        "Password");

    const string excelFileName = @"C:\sample.xls";

    var fileItem = await connection.Items.Root.NewFile("ExcelFile",
        StiFileType.Excel).SaveAsync();
    await fileItem.UploadFromFileAsync(excelFileName);

    var table1 = await fileItem.GetDataAsync("Sheet1");
    var firstCellData = table1.Rows[0].ItemArray[0];

    await connection.Accounts.Users.LogoutAsync();
}
...
```

1.2 REST API

REST API allows you quick and easy access to the main functions of **Stimulsoft Server** and automates user actions of the system in your application. Access points to the REST-services should be on your domain and be accessible via HTTP/HTTPS. For example, we will use the locally installed version of Stimulsoft Server with the URL **<http://reports.stimulsoft.com>**. A relative path begins with the prefix `/1/` which means the first version of REST API.

EndPoint	HTTP Verb	Action
/1/login	GET	Login registered user
/1/logout	DELETE	User logout
/1/signup	POST	New user registration
/1/users	GET	Getting a list of users
/1/users	POST	Creating a new user
/1/users/<UserId>	GET	Getting information about

		the user <UserId>
/1/users/<UserId>	PUT	Changing user information <UserId>
/1/users/<UserId>	DELETE	Removing a user <UserId>
/1/users/current	GET	Getting information about the current user
/1/users/current	PUT	Changing the current user information
/1/users/<UserId>/changepassword	PUT	Change user password
/1/users/current/changepassword	PUT	Changing the password of the current user
/1/users/<UserId>/resetpassword	PUT	Initialize reset the user's <UserId> password
/1/users/current/resetpassword	PUT	Initialize the current user password reset
/1/resetpassword/secretcode	PUT	Confirm password reset by a secret code
/1/roles	GET	Getting a list of roles
/1/roles/<RoleId>	GET	Getting information about the role <RoleId>
/1/roles	POST	Creating a new role
/1/roles/<RoleId>	PUT	Changing information about the role <RoleId>
/1/roles/<RoleId>	DELETE	Remove a role <RoleId>
/1/items	GET	Getting a list of items
/1/items/<ItemId>	GET	Getting information about the element <ItemId>
/1/items	POST	Create a new item
/1/items/<ItemId>	PUT	Changing information

		about the element <ItemId>
/1/items/<ItemId>	DELETE	Deleting an element <ItemId>
/1/items/<ItemId>/share	GET	Getting information about public access to the element <ItemId>
/1/items/<ItemId>/share	PUT	Changing data on public access to the element <ItemId>
/1/items/<ItemId>/share	DELETE	Removing an element <ItemId> from public access
/1/schedulers	GET	Getting a list of schedulers
/1/schedulers/<SchedulerId>	GET	Getting information about the scheduler <SchedulerId>
/1/schedulers	POST	Creating a new scheduler
/1/schedulers/<SchedulerId>	PUT	Changing information about the scheduler <SchedulerId>
/1/schedulers/<SchedulerId>	DELETE	Removal of scheduler <SchedulerId>
/1/schedulers/<SchedulerId>/status	GET	Getting information about the state of the scheduler <SchedulerId>
/1/schedulers/<SchedulerId>/status	PUT	Setting state of scheduler <SchedulerId>
/1/schedulers/<SchedulerId>/run	PUT	Immediate start of scheduler <SchedulerId>
/1/files	GET	Getting a list of files
/1/files/<FileId>/	GET	Downloading a file

		<FileId>
/1/files	POST	Creating a new file
/1/files/<FileId>/	PUT	Appending a new chunk to file <FileId>
/1/files/<FileId>/	DELETE	Removing a file <FileId>
/1/reporttemplates	GET	Getting a list of report templates
/1/reporttemplates/<ReportTemplateId>/	GET	Getting information about the report template <ReportTemplateId>
/1/reporttemplates	POST	Creating a new report template
/1/reporttemplates/<ReportTemplateId>/	DELETE	Removing a report template <ReportTemplateId>
/1/reporttemplates/<ReportTemplateId>/run	PUT	Build the report template and save result to another item
/1/reporttemplates/<ReportTemplateId>/duplicate	POST	Creating a new copy of the report template in a workspace of the logged-in user.
/1/reportsnapshots	GET	Getting a list of report snapshots
/1/reportsnapshots/<ReportSnapshotId>/	GET	Getting information about the report snapshot <ReportSnapshotId>
/1/reportsnapshots	POST	Creating a new report snapshot
/1/reportsnapshots/<ReportSnapshotId>/	DELETE	Removing a report snapshot <ReportSnapshotId>

/1/reportsnapshots/ <ReportSnapshotId>/ export	PUT	Export data from report snapshot
--	-----	--

1.2.1 Object Model

The input data for each command are custom HTTP-headers prefixed with "x-sti-" and described in the table below:

Custom header	Description
x-sti-sessionkey	The current session key is issued after successful authentication, used in almost all commands
x-sti-workspacekey	Key of the current workspace, used to limit the sample users and roles
x-sti-versionkey	Key of the last chunk of the uploaded file
x-sti-username	Username for authentication
x-sti-password	Password to login
x-sti-newpassword	The new user's password, used to change the password and reset password
x-sti-currentpassword	User's current password, used to change the password
x-sti-index	Index of the element to which you want to fetch, used to limit the data fetch
x-sti-count	Number of items that need to get, used to limit the data fetch
x-sti-allowdeleted	Determines whether the elements in the fetch display deleted, default is false
x-sti-itemkey	The key of element, which is the work, used to limit the data fetch
x-sti-filterident	Filtering of the elements by identifiers,

	used to limit the data fetch
x-sti-status	Task scheduler status, used to manage the schedulers
x-sti-allowmovetorecyclebin	Allows deleting an item to the recycle bin, by default it is set to true
x-sti-destinationitemkey	Allows indicates an item that will be stored data

Other information transmitted in the body of the request in the JSON format. Permitted to transfer the input parameters in the request body as a simple JSON-object with fields whose names match the custom headers, but lack the prefix "x-sti-". Parameter names are not case sensitive, regardless of how they transfer (custom headers or POST-data). As the output data is used JSON-object for all commands. Even in the case of error or successful completion of an object returned, indicating success of query processing server.

Error example:

Sample **JSON** response

```
...
{
  "Ident": "UserFetchAll",
  "ResultNotice": {
    "Ident": "IsNotSpecified",
    "Arguments": [
      "SessionKey"
    ]
  }
}
...
```

Another error example:

Sample **JSON** response

```
...
{
  "Ident": "UserSave",
  "ResultNotice": {
    "Ident": "AccessDenied",
    "CustomMessage": "You should specify the CurrentPassword argument when you want to change the user password!"
  }
}
```

```
}  
}  
...
```

Example of successful command execution, not returning data:

Sample **JSON** response

```
...  
{  
  "Ident": "UserLogout",  
  "ResultSuccess": true  
}  
...
```

Obviously, if the command does not return HTTP error and successfully passed the server to perform, any response with JSON-object will have an HTTP-status "200 OK". In this case, success of command execution needs to be checked on a condition of the field `ResultSuccess` and availability of data in the field `ResultNotice`. If `ResultNotice` field is not empty, then the command has problems that are described nested object contained in this field.

1.2.2 Sign Up

Description:

Create a new user (in new workspace)

Url Structure:

<http://reports.stimulsoft.com/1/signup>

Method:

POST

Parameters:

In POST-data must specify the JSON-object describing a new user:

POST-data in the **JSON**-object

```
...  
{  
  'FirstName': 'John',  
  'LastName': 'Doe',  
  'UserName': 'j@d.com',  
  'Password': '111111'  
}
```

```
}  
...
```

CURL example:

```
curl -X POST -d "{ 'FirstName': 'John', 'LastName': 'Doe', 'UserName': 'j
```

Returns:

The JSON object containing the field `ResultUserKey` with the key of the new user. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...  
{  
  "Ident": "UserSignUp",  
  "ResultUserKey": "ddb025415b68494ca8a9aee27ad73bc1",  
  "ResultSuccess": true  
}  
...
```

1.2.3 Login

Description:

Login with username and password.

Url Structure:

```
http://reports.stimulsoft.com/1/login
```

Method:

GET

Parameters:

Two custom header: `x-sti-UserName` and `x-sti-Password`, containing the username and password, respectively.

CURL example:

```
curl -X GET -H "x-sti-UserName: a@a.com" -H "x-sti-Password: 111111" http
```

Returns:

The JSON object containing the field `ResultSessionKey`, which will be further used for communication with the server. The success of the command execution is

checked by the content of the field `ResultSuccess`.

Sample **JSON** response

```
...
{
  "Ident": "UserLogin",
  "ResultSessionKey": "1716cdf3dd2b480580824798a03f030d",
  "ResultWorkspaceKey": "1b11a087888f4a968ecbbaf74423647c",
  "ResultUserKey": "50a4f98ec1804a6498829b05554c5608",
  "ResultRole": {
    "Name": "Supervisor",
    "Created": "\\Date(1426675381267)\\",
    "Modified": "\\Date(1426675381267)\\",
    "Permissions": {
      "ItemCalendars": "CreateDeleteModifyView",
      "ItemCloudStorages": "CreateDeleteModifyView",
      "ItemContactLists": "CreateDeleteModifyView",
      "ItemDashboards": "All",
      "ItemDataSources": "All",
      "ItemFiles": "CreateDeleteModifyView",
      "ItemFolders": "CreateDeleteModifyView",
      "ItemReportSnapshots": "All",
      "ItemReportTemplates": "All",
      "ItemSchedulers": "All"
    },
    "IsSupervisor": true,
    "IsAdministrator": true,
    "IsSystem": true,
    "Key": "Supervisor"
  },
  "ResultSettings": {
    "Localization": "en",
    "Theme": {
      "Ident": "Office2013White",
      "Style": "Teal"
    },
    "Key": "50a4f98ec1804a6498829b05554c5608"
  },
  "ResultProductInfo": {
    "ProductName": "Stimulsoft Reports Server"
  },
  "ResultSuccess": true
}
...
```

1.2.4 Logout

Description:

Log out of the current user.

Url Structure:

<http://reports.stimulsoft.com/1/logout>

Method:`DELETE`**Parameters:**

A custom header `x-sti-SessionKey` contains the session key of the current user.

CURL example:

```
curl -X DELETE -H "x-sti-SessionKey: 1638c5f0e7d347dabb7651f194768a7e" ht
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...
{
  "Ident": "UserLogout",
  "ResultSuccess": true
}
...
```

1.2.5 Users

For obtaining the list of users, data modification, as well as to create new users in the current workspace and delete users, use the command `Users` with different methods.

Name	Description
GET List	Getting a list of users in a workspace of the logged-in user.
GET Info	Getting information about the user in a workspace of the logged-in user.
POST Create	Creating a new user in a workspace of the logged-in user.
PUT Edit	Changing user data in a workspace of the logged-in user. This command does not allow change of the unique user name, which is used as an

	identifier, and a password - for this purpose there is other command.
DELETE	Removing a user from the current workspace. Removing a last user with administrator privileges isn't allowed.

Managing user passwords requires a special approach to security, so to change and reset the password using a number of separate commands. Changing the password requires an existing password. Reset password occurs in two stages - a password reset request and execution of the procedure. Password reset request executed with the command `resetpassword`, at the same time to the email address specified in the user data sent to e-mail with a link, activating the second stage - the actual password reset. To activate the second stage through REST-interface, you must run command `resetpassword` with the indication a secret one-time code (specified in the link sent in an email message), valid for two hours. Next to the postal address of the user sent another message indicating a new password:

Name	Description
Change password	Change user password.
Reset password	Activating of password reset. At this stage, checked the security code, and if it is correct, the password is reset to the new specified in the parameters. To activate the procedure requires new password and the secret code obtained at the first stage. Executing this command does not require a session key.

1.2.5.1 GET List

Description:

Getting a list of users in a workspace of the logged-in user.

Url Structure:

`http://reports.stimulsoft.com/1/users`

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. You may use header `x-sti-WorkspaceKey`, containing key workspace that you are requesting a list of users.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: 55397673d1604f3a9aabbf57c4ebaf856" http:
```

Returns:

The JSON object containing the field `ResultSessionKey`, which is a list of members of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "UserFetchAll",
  "ResultUsers": [
    {
      "RoleKey": "Administrator",
      "WorkspaceKey": "c25a38229d484ce589b983ed680f681e",
      "FirstName": "John",
      "LastName": "Doe",
      "UserName": "j@d.com",
      "Created": "\\Date(1424679345637)\\",
      "Modified": "\\Date(1424679345637)\\",
      "LastLogin": "\\Date(1424698541377)\\",
      "Key": "ddb025415b68494ca8a9aee27ad73bc1"
    },
    {
      "RoleKey": "User",
      "WorkspaceKey": "c25a38229d484ce589b983ed680f681e",
      "FirstName": "John5",
      "LastName": "Doe5",
      "UserName": "j5@d5.com",
      "Created": "\\Date(1424699430000)\\",
      "Modified": "\\Date(1424699430367)\\",
      "Key": "e7c2b7d3160f418b9286fbb24e7b0dd9"
    }
  ],
  "ResultSuccess": true
}
...
```

1.2.5.2 GET Info

Description:

Getting information about the user in a workspace of the logged-in user.

Url Structure:

`http://reports.stimulsoft.com/1/users/username`

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The username parameter in the URI is the key user or its name and indicates the user whose data you want to get. Instead of the user name or key, you can specify the keyword "current" which replaces the identifier of the current user.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: ea8cc765d54241e18347a043e187ada3" http:
```

Returns:

The JSON object containing the field `ResultUsers`, in which there is data on the user in the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample **JSON** response

```
...
{
  "Ident": "UserGet",
  "ResultUser": {
    "RoleKey": "User",
    "WorkspaceKey": "c25a38229d484ce589b983ed680f681e",
    "FirstName": "John51",
    "LastName": "Doe51",
    "UserName": "j51@d51.com",
    "Created": "\\Date(1424765701477)\\",
    "Modified": "\\Date(1424765701777)\\",
    "Key": "029f450a853b4248bdc278c382512f90"
  },
  "ResultSuccess": true
}
...
```

1.2.5.3 POST Create

Description:

Creating a new user in a workspace of the logged-in user.

Url Structure:

`http://reports.stimulsoft.com/1/users`

Method:

POST

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. In POST-data must specify the JSON-object describing the new user:

POST-data in the JSON-object

```
...
{
  'FirstName': 'John5',
  'LastName': 'Doe5',
  'UserName': 'j5@d5.com',
  'Password': '111111'
}
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: 55397673d1604f3a9aabf57c4ebaf856" -d "
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...
{
  "Ident": "UserSave",
  "ResultSuccess": true
}
...
```

1.2.5.4 PUT Edit

Description:

Changing user data in a workspace of the logged-in user. This command does not allow change of the unique user name, which is used as an identifier, and a password - for this purpose there is other command.

Url Structure:

```
http://reports.stimulsoft.com/1/users/username
```

Method:

```
PUT
```

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The username parameter in the URI is the key user or its name and indicates the user whose data you want to edit. Instead of the user name or key, you can specify the keyword "current" which replaces the identifier of the current user. In POST-data must specify the JSON-object describing the changed user data:

POST-data in the JSON-object

```
...
{
  'FirstName': 'John00',
  'LastName': 'Doe00'
}
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ea8cc765d54241e18347a043e187ada3" -d "{
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...
{
  "Ident": "UserSave",
  "ResultSuccess": true
}
...
```

1.2.5.5 DELETE

Description:

Removing a user from the current workspace. Removing a last user with administrator privileges isn't allowed.

Url Structure:

`http://reports.stimulsoft.com/1/users/username`

Method:

DELETE

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The username parameter in the URI is the key of user or its name and indicates the user whose data you want to delete.

CURL example:

```
curl -X DELETE -H "x-sti-SessionKey: ea8cc765d54241e18347a043e187ada3" ht
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...
{
  "Ident": "UserDelete",
  "ResultSuccess": true
}
...
```

1.2.5.6 Change password

Description:

Change user password.

Url Structure:

`http://reports.stimulsoft.com/1/users/username/changepassword`

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The username parameter in the URI is the key user or its name and indicates the user whose data you want to get. Instead of the user name or key, you can specify the keyword "current" which replaces the identifier of the current user. In POST-data must specify the JSON-object describing the current and new passwords:

POST-data in the JSON-object

```
...
{
  'CurrentPassword': '111111',
  'NewPassword': '222222'
}
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ea8cc765d54241e18347a043e187ada3" -d "{
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...
{
  "Ident": "UserChangePassword",
  "ResultSuccess": true
}
...
```

1.2.5.7 Reset password**Description:**

Activating of password reset. At this stage, checked the security code, and if it is correct, the password is reset to the new specified in the parameters. To activate the procedure requires new password and the secret code obtained at the first stage. Executing this command does not require a session key.

Url Structure:

`http://reports.stimulsoft.com/1/resetpassword/secretcode`

Method:

PUT

Parameters:

A custom header `x-sti-NewPassword` contains the new password. The `secretcode` parameter in the URI is the secret code obtained in the first stage of the procedure a password reset, and indicates the user whose password you want to reset. The secret code is valid for two hours from the time of the query reset your password.

CURL example:

```
curl -X PUT -H "x-sti-NewPassword: 222222" -d "" http://reports.stimulsoft.com/1/resetpassword/secretcode
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...
{
  "Ident": "UserResetPasswordComplete",
  "ResultUserName": "j@d.com",
  "ResultSuccess": true
}
...
```

1.2.6 Roles

Object Role describes permissions for different data types supported by the system. The main characteristic of this object is set `Permissions`, which may contain a table of permissions:

Permission	Description
ItemCalendars	Permissions for Calendars
ItemCloudStorages	Permissions for Cloud Storages
ItemContactLists	Permissions for Contactlists
ItemDashboards	Permissions for Dashboards

ItemDataSources	Permissions for Datasources
ItemFiles	Permissions for Files
ItemFolders	Permissions for Folder
ItemReportSnapshots	Permissions for ReportSnapshots
ItemReportTemplates	Permissions for ReportTemplates
ItemSchedulers	Permissions for Schedulers

Each of these permissions have one of the values:

Value	Description
None	Deny all
Create	Allows to create an item
Delete	Allows to delete an item
Modify	Allows to modify an item
Run	Allows to run an item
View	Allows to view an item
DeleteModifyView	Allows delete, modify and view an item
ModifyView	Allows modify and view an item
CreateDeleteModifyView	Allows create, delete, modify and view an item
RunView	Allows run and view an item
All	Allow any action with an item

Permission example:

Sample **JSON** response

```
...
"Permissions": {
  "ItemCalendars": "CreateDeleteModifyView",
  "ItemCloudStorages": "CreateDeleteModifyView",
  "ItemContactLists": "CreateDeleteModifyView",
  "ItemDashboards": "All",
  "ItemDataSources": "CreateDeleteModifyView",
  "ItemFiles": "CreateDeleteModifyView",
  "ItemFolders": "CreateDeleteModifyView",
  "ItemReportSnapshots": "All",
  "ItemReportTemplates": "All",
  "ItemSchedulers": "All"
}
...
```

To get the list of roles, data modification, and to create new roles in the current workspace and delete existing roles, use the command Roles with different methods. The default configuration of the system, there are four roles (fields marked with "IsSystem": true) - Supervisor, Administrator, Manager and User (a role name the same as the key). These roles can't be removed or changed. It is possible to modify only the roles created by the user.

Name	Description
GET List	Getting a list of roles.
GET Info	Getting information about the role in a workspace of the logged-in user.
POST Create	Creating a new role in a workspace of the logged-in user. This command ignores the initialization key of role (field Key) and puts this value automatically.
PUT Edit	Changing roles of users in a workspace of the logged-in user. This command does not allow a change of the unique role name (field Key).
DELETE	Removing a role from the current workspace. Removing a last role with administrator privileges isn't allowed.

1.2.6.1 GET List

Description:

Getting a list of roles.

Url Structure:

`http://reports.stimulsoft.com/1/roles`

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. You may use header `x-sti-WorkspaceKey`, containing key workspace that you are requesting a list of roles.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: ea8cc765d54241e18347a043e187ada3" http:
```

Returns:

The JSON object containing the collection `ResultUsers`, which is a list of the roles of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "RoleFetchAll",
  "ResultRoles": [
    {
      "Name": "Supervisor",
      "Created": "\\Date(1425645969540)\\",
      "Modified": "\\Date(1425645969540)\\",
      "Permissions": {
        "ItemCalendars": "CreateDeleteModifyView",
        "ItemCloudStorages": "CreateDeleteModifyView",
        "ItemContactLists": "CreateDeleteModifyView",
        "ItemDashboards": "All",
        "ItemDataSources": "CreateDeleteModifyView",
        "ItemFiles": "CreateDeleteModifyView",
        "ItemFolders": "CreateDeleteModifyView",
        "ItemReportSnapshots": "All",
        "ItemReportTemplates": "All",
        "ItemSchedulers": "All"
      },
      "IsSupervisor": true,
    }
  ]
}
```

```
"IsAdministrator": true,
"IsSystem": true,
"Key": "Supervisor"
},
{
  "Name": "Administrator",
  "Created": "\\Date(1425645969540)\\",
  "Modified": "\\Date(1425645969540)\\",
  "Permissions": {
    "ItemCalendars": "CreateDeleteModifyView",
    "ItemCloudStorages": "CreateDeleteModifyView",
    "ItemContactLists": "CreateDeleteModifyView",
    "ItemDashboards": "All",
    "ItemDataSources": "CreateDeleteModifyView",
    "ItemFiles": "CreateDeleteModifyView",
    "ItemFolders": "CreateDeleteModifyView",
    "ItemReportSnapshots": "All",
    "ItemReportTemplates": "All",
    "ItemSchedulers": "All"
  },
  "IsAdministrator": true,
  "IsSystem": true,
  "Key": "Administrator"
},
{
  "Name": "Manager",
  "Created": "\\Date(1425645969540)\\",
  "Modified": "\\Date(1425645969540)\\",
  "Permissions": {
    "ItemCalendars": "CreateDeleteModifyView",
    "ItemCloudStorages": "CreateDeleteModifyView",
    "ItemContactLists": "CreateDeleteModifyView",
    "ItemDashboards": "All",
    "ItemDataSources": "CreateDeleteModifyView",
    "ItemFiles": "CreateDeleteModifyView",
    "ItemFolders": "CreateDeleteModifyView",
    "ItemReportSnapshots": "All",
    "ItemReportTemplates": "All",
    "ItemSchedulers": "View"
  },
  "IsSystem": true,
  "Key": "Manager"
},
{
  "Name": "User",
  "Created": "\\Date(1425645969540)\\",
  "Modified": "\\Date(1425645969540)\\",
  "Permissions": {
    "ItemCalendars": "View",
    "ItemCloudStorages": "View",
    "ItemContactLists": "View",
    "ItemDashboards": "RunView",
    "ItemDataSources": "View",
    "ItemFiles": "View",
    "ItemFolders": "View",
```

```
        "ItemReportSnapshots": "RunView",
        "ItemReportTemplates": "RunView"
    },
    "IsSystem": true,
    "Key": "User"
}
],
"ResultSuccess": true
}
...
```

1.2.6.2 GET Info

Description:

Getting information about the role in a workspace of the logged-in user.

Url Structure:

<http://reports.stimulsoft.com/1/roles/rolename>

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The rolename parameter in the URI is the key of role and indicates the role whose data you want to get.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: ea8cc765d54241e18347a043e187ada3" -d ""
```

Returns:

The JSON object containing the field `ResultRole`, which is the desired role of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "RoleGet",
  "ResultRole": {
    "Name": "User",
    "Created": "\\Date(1425646886701)\\",
    "Modified": "\\Date(1425646886701)\\",
    "Permissions": {
      "ItemCalendars": "View",
```

```
    "ItemCloudStorages": "View",
    "ItemContactLists": "View",
    "ItemDashboards": "RunView",
    "ItemDataSources": "View",
    "ItemFiles": "View",
    "ItemFolders": "View",
    "ItemReportSnapshots": "RunView",
    "ItemReportTemplates": "RunView"
  },
  "IsSystem": true,
  "Key": "User"
},
"ResultSuccess": true
}
...
```

1.2.6.3 POST Create

Description:

Creating a new role in a workspace of the logged-in user. This command ignores the initialization key of role (field Key) and puts this value automatically.

Url Structure:

<http://reports.stimulsoft.com/1/roles>

Method:

POST

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. In POST-data must specify the JSON-object describing the new role:

POST-data in the JSON-object

```
...
{
  'Name': 'TestRole',
  'Created': '\Date(1424872039434)\',
  'Modified': '\Date(1424872039434)\',
  'Permissions':
  {
    'ItemCalendars': 'All',
    'ItemCloudStorages': 'View',
    'ItemContactLists': 'View',
    'ItemDashboards': 'View',
    'ItemDataSources': 'View',
    'ItemFiles': 'View',
    'ItemFolders': 'View',
    'ItemReportSnapshots': 'RunView',

```

```
'ItemReportTemplates': 'RunView'
}
}
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: ea8cc765d54241e18347a043e187ada3" -d "{
'ItemCalendars': 'All', 'ItemCloudStorages': 'View', 'ItemContactLists': 'Vie
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...
{
  "Ident": "RoleSave",
  "ResultSuccess": true
}
...
```

1.2.6.4 PUT Edit**Description:**

Changing roles of users in a workspace of the logged-in user. This command does not allow a change of the unique role name (field Key).

Url Structure:

`http://reports.stimulsoft.com/1/roles/rolename`

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The rolename parameter in the URI is the key of role and indicates the role whose data you want to change. In POST-data must specify the JSON-object describing the changing role:

POST-data in the JSON-object

```
...
```

```
{
  'Name': 'TestRole',
  'Permissions':
  {
    'ItemCalendars': 'All',
    'ItemReportTemplates': 'View'
  }
}
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ea8cc765d54241e18347a043e187ada3" -d "{
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...
{
  "Ident": "RoleSave",
  "ResultSuccess": true
}
...
```

1.2.6.5 DELETE**Description:**

Removing a role from the current workspace. Removing a last role with administrator privileges isn't allowed.

Url Structure:

`http://reports.stimulsoft.com/1/roles/rolename`

Method:

`DELETE`

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The rolename parameter in the URI is the key of role and indicates the role whose data you want to delete.

CURL example:

```
curl -X DELETE -H "x-sti-SessionKey: ea8cc765d54241e18347a043e187ada3" ht
```

Returns:

The JSON-object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample **JSON** response

```
...
{
  "Ident": "RoleDelete",
  "ResultSuccess": true
}
...
```

1.2.7 Items

Items object describes the data items supported by the system. This is an abstract class that combines different types of elements. At creation of an element its type is specified in the `Ident` field and never changes:

Object	Description	Ident
<code>StiCalendarItem</code>	Calendar, used to create schedules.	<code>CalendarItem</code>
<code>StiCloudFileItem</code>	Cloud file, used to connect to external data files.	<code>CloudFileItem</code>
<code>StiCloudStorageItem</code>	Cloud storage, used to create directories for storing the contents of cloud items.	<code>CloudStorageItem</code>
<code>StiContactListItem</code>	Contact list, used to send data via email.	<code>ContactListItem</code>
<code>StiDashboardItem</code>	Dashboard template.	<code>DashboardItem</code>
<code>StiDataQueryItem</code>	Data Query, used to retrieve data from SQL data sources.	<code>DataQueryItem</code>
<code>StiDataRelationItem</code>	Data Relation, used to join data sources.	<code>DataRelationItem</code>

StiDataSnapshotItem	Data Snapshot, used to store a point-in-time copy of a dataset	DataSnapshotItem
StiDataSourceItem	Data source, used to connect to external data	DataSourceItem
StiDataTableItem	Data table, used to structure data from data sources.	DataTableItem
StiExcelTableItem	Excel table, used to structure data from Excel data sources.	ExcelTableItem
StiFileItem	File, used to connect to external data files.	FileItem
StiFileStorageItem	File storage, used to create directories for storing the contents of server items.	FileStorageItem
StiFolderItem	Folder, provides a hierarchical structure for organizing elements.	FolderItem
StiReportSnapshotItem	Report snapshot, a rendered report containing data.	ReportSnapshotItem
StiReportTemplateItem	Report template.	ReportTemplateItem
StiSchedulerItem	Scheduler, used to automate actions.	SchedulerItem
StiMailingItem	Mailing, used to automate email distribution.	MailingItem
StiLinkItem	Link, used to create a shortcut to a report or dashboard.	LinkItem

To get the list of elements, modify the data, and to create new elements in the

current workspace and removing existing elements, use command Items with various methods. Each element has a unique key, which uniquely identifies it in the list of elements. Data hierarchy as a tree is realized by an element type `StiFolderItem`, key element it is specified in the field of `FolderKey` as designation of the parental folder of an element. The identifier of this element is specified in the `FolderKey` of other elements and provides identification of the parent folder of an element. If this field is empty or not initialized to any value, then the element belongs to the root folder.

Name	Description
GET List	Getting a list of elements in a workspace of the logged-in user. The list is returned to the specified folder.
GET Info	Getting information about the element in a workspace of the logged-in user.
POST Create	Creating a new element in a workspace of the logged-in user. To successfully run the command you must fill in the fields <code>Ident</code> (assigned a value in accordance with the required type of item has one of the values listed in the table above), and <code>Name</code> . <code>FolderKey</code> field may contain the key of the parent folder, making sure that the element in the tree. If <code>FolderKey</code> empty or not specified, the item is displayed in the root folder.
PUT Edit	Changing element in a workspace of the logged-in user. This command does not allow change of the unique item key, which is used as an identifier (field <code>Key</code>), and the type (field <code>Ident</code>). Changing field <code>FolderKey</code> can move an item to another folder in the tree (the root, if you specify a null value).

[DELETE](#)

Removing an element from the current workspace. However, the use of this command does not guarantee the immediate removal of the element from a tree, because the command only creates an internal task of server to delete the item and the actual deletion may be delayed for some time.

1.2.7.1 GET List

Description:

Getting a list of elements in a workspace of the logged-in user. The list is returned to the specified folder.

Url Structure:

`http://reports.stimulsoft.com/1/items`

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. Custom header `x-sti-ItemKey` used to identify the parent folder, which list of elements needs to be retrieved. If this header is not present, it will get a list of elements of the root folder. To filter element types used header `x-sti-FilterIdent`. It may contain one of the values in the table above `Ident`. If this header is absent, all elements from the requested collection will be returned. It is also possible to use the header `x-sti-AllowDeleted`, which is responsible for displaying the elements placed in the recycle bin (not removed completely).

CURL example:

```
curl -X GET -H "x-sti-SessionKey: ea8cc765d54241e18347a043e187ada3" -H "x
```

Returns:

The JSON object containing the collection `ResultItems`, which contains a list of items in the specified folder of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample **JSON** response

```

...
{
  "Ident": "ItemFetchAll",
  "ResultItems": [
    {
      "Ident": "FolderItem",
      "FolderKey": "7800e3265d06418a9ac4feb977fd4040",
      "WorkspaceKey": "c25a38229d484ce589b983ed680f681e",
      "Name": "InternalFolder",
      "Description": "This is a second level folder",
      "Created": "\\Date(1425024762360)\\",
      "Modified": "\\Date(1425024762360)\\",
      "IsMoveable": true,
      "Key": "ac9485530c2e42cf9edef840a4816c4f"
    },
    {
      "Ident": "ReportTemplateItem",
      "StateKey": "2",
      "FolderKey": "7800e3265d06418a9ac4feb977fd4040",
      "WorkspaceKey": "c25a38229d484ce589b983ed680f681e",
      "Name": "SecondEmptyReport",
      "Description": "This is a internal report",
      "Created": "\\Date(1425025029473)\\",
      "Modified": "\\Date(1425042737867)\\",
      "IsMoveable": true,
      "Key": "649f1ff97b21448d963c4747279e86d9"
    }
  ],
  "ResultSuccess": true
}
...

```

Table of Element Types Used in x-sti-FilterIdent Header

Name	Description
Sample	"x-sti-FilterIdent: ReportTemplateItem"
CalendarItem	This provides the ability to return a list of calendar items.
CloudFileItem	This provides the ability to return a list of cloud file items.
CloudStorageItem	This provides the ability to return a list of cloud storage items.

ContactListItem	This provides the ability to return a list of contact list items.
DashboardItem	This provides the ability to return a list of dashboard items.
DataQueryItem	This provides the ability to return a list of data query items.
DataRelationItem	This provides the ability to return a list of data relation items.
DataSnapshotItem	This provides the ability to return a list of data snapshot items.
DataSourceItem	This provides the ability to return a list of data source items.
DataTableItem	This provides the ability to return a list of data table items.
ExcelTableItem	This provides the ability to return a list of Excel table items.
FileItem	This provides the ability to return a list of file items.
FileStorageItem	This provides the ability to return a list of file storage items.
FolderItem	This provides the ability to return a list of folder items.
ReportSnapshotItem	This provides the ability to return a list of report snapshot items.
ReportTemplateItem	This provides the ability to return a list of report template items.
SchedulerItem	This provides the ability to return a list of scheduler items.
MailingItem	This provides the ability to return a list of mailing items.
LinkItem	This provides the ability to return a list of link items.

1.2.7.2 GET Info

Description:

Getting information about the element in a workspace of the logged-in user.

Url Structure:

`http://reports.stimulsoft.com/1/items/itemkey`

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The `itemkey` parameter in the URI is the key of the element and indicates the element whose data you want to get.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: da5053abac4544e9856e05bbda14f46a" http:
```

Returns:

The JSON-object containing the field `ResultItem`, which is the required element of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "ItemGet",
  "ResultItem": {
    "Ident": "FileItem",
    "FileType": "Xml",
    "AttachedItems": [
      "fa3514a207504deea0c065032d5d438f"
    ],
    "IsDataSource": true,
    "FolderKey": "d1a339068a474eaab65628f2fbef33a6",
    "WorkspaceKey": "1b11a087888f4a968ecbbaf74423647c",
    "Name": "Demo.xml",
    "Description": "",
    "Created": "\\Date(1427455735000)\\",
    "Modified": "\\Date(1427986596000)\\",
    "IsMoveable": true,
    "Key": "d2283e85e9724859bcd024c3f7b982ea"
  },
  "ResultLastVersionKey": "ab832999641b458eacd8969a408e303e",
  "ResultSuccess": true
}
```

```
}  
...
```

1.2.7.3 GET Versions

Description:

Gets the version history of a specified item in the workspace of the logged-in user.

Url Structure:

`http://reports.stimulsoft.com/1/items/{itemKey}/versions`

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The item identifier is specified in the URL path parameter `{itemKey}`.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: 09267e7288e44f37a0b376e5841b19ff" http:
```

Returns:

A JSON object containing the `ResultVersions` collection, which represents the list of available versions for the specified item. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...  
{  
  "Ident": "ItemFetchVersions",  
  "AllowDeleted": false,  
  "ResultVersions": [  
    {  
      "StateKey": "1",  
      "ItemKey": "f10358b8a3a44a31aa3690df8c4b1c95",  
      "WorkspaceKey": "e15516cd7d6d4f30973f7856dfb7b6d8",  
      "Created": "\/Date(1782200646987)\/",  
      "UserName": "a a",  
      "Key": "ea7314336e374a88a52c03b25245796c"  
    },  
    {  
      "StateKey": "2",  
      "ItemKey": "f10458b8a3a44a31aa3690df8c4b1c95",  
      "WorkspaceKey": "e11216cd7d6d4f30973f7856dfb7b6d8",  
      "Created": "\/Date(1782200646981)\/",  
    }  
  ]  
}
```

```
        "UserName": "b b",  
        "Key": "ea7314336e374a88a52c03b25245356c"  
    },  
    ],  
    "ResultSuccess": true  
}  
...
```

1.2.7.4 POST Create

Description:

Creating a new element in a workspace of the logged-in user. To successfully run the command you must fill in the fields `Ident` (assigned a value in accordance with the required type of item has one of the values listed in the table above), and `Name`. `FolderKey` field may contain the key of the parent folder, making sure that the element in the tree. If `FolderKey` empty or not specified, the item is displayed in the root folder.

Url Structure:

`http://reports.stimulsoft.com/1/items`

Method:

POST

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. In POST-data must specify the JSON-object, which describing the new element:

POST-data in the JSON-object

```
...  
{  
  'Ident': 'FolderItem',  
  'Name': 'InternalFolder',  
  'Description': 'This is a second level folder',  
  'FolderKey': '7800e3265d06418a9ac4feb977fd4040'  
}  
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: ea8cc765d54241e18347a043e187ada3" -d "
```

Returns:

The JSON-object containing the field `ResultSuccess` which indicates that the

command is executed successfully.

Sample **JSON** response

```
...
{
  "Ident": "ItemSave",
  "ResultSuccess": true
}
...
```

1.2.7.5 PUT Edit

Description:

Changing element in a workspace of the logged-in user. This command does not allow change of the unique `item key`, which is used as an identifier (field `Key`), and the type (field `Ident`). Changing field `FolderKey` can move an item to another folder in the tree (the root, if you specify a null value).

Url Structure:

`http://reports.stimulsoft.com/1/items/itemkey`

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The `itemkey` parameter in the URI is the key of item and indicates the element whose data you want to change. In POST-data must specify the JSON-object describing the changed item data:

POST-data in the **JSON**-object

```
...
{
  'Name': 'SecondEmptyReport',
  'Description': 'This is a edited report',
  'FolderKey': ''
}
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ea8cc765d54241e18347a043e187ada3" -d "{
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample **JSON** response

```
...
{
  "Ident": "ItemSave",
  "ResultSuccess": true
}
...
```

1.2.7.6 PUT Restore

Description:

Restores an item from the RecycleBin to its original location in the workspace of the currently logged-in user.

Url Structure:

`http://cloud.stimulsoft.com/1/items/itemKey/restore`

Method:

`PUT`

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The itemkey parameter in the URI is the key of item and indicates the element whose data you want to change.

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: 09267e7288e44f37a0b376e5841b19ff" -d ""
```

Returns:

Returns a JSON object containing the `ResultSuccess` field, which indicates whether the item was successfully restored.

Sample **JSON** response

```
...
{
  "Ident": "ItemRestore",
  "AllowCreateWaitNotification": true,
  "AllowNotifications": false,
  "AllowSchedulerSignals": false,
  "AllowSignals": false,
}
```

```
"AllowRemovingSource": false,  
"NotificationVisibility": true,  
"ResultTaskKey": "878f274f95d040cabfe1079a0a4148f1",  
"ResultSuccess": true  
}  
...
```

1.2.7.7 DELETE

Description:

Removing an element from the current workspace. However, the use of this command does not guarantee the immediate removal of the element from a tree, because the command only creates an internal task of server to delete the item and the actual deletion may be delayed for some time.

Url Structure:

`http://reports.stimulsoft.com/1/items/itemkey`

Method:

DELETE

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-AllowMoveToRecycleBin` allows deleting an item to the recycle bin, by default it is set to true. To remove an item with referenced resources set `x-sti-AllowMoveToRecycleBin` to false. The `itemkey` parameter in the URI is the key of item and indicates the item whose data you want to delete.

CURL example:

```
curl -X DELETE -H "x-sti-SessionKey: ea8cc765d54241e18347a043e187ada3" -H
```

Returns:

The success of the command execution is checked by the content of the field `ResultSuccess`. `ResultTaskKey` field contains unique key of internal server tasks, created to remove the item.

Sample JSON response

```
...  
{  
  "Ident": "ItemDelete",  
  "ResultTaskKey": "3c1df5fa46a14a17af32a5259834e254",  
  "ResultSuccess": true  
}
```

...

1.2.7.8 Share

Some elements of the navigator tree (`StiFileItem`, `StiReportSnapshotItem`, `StiReportTemplateItem`), can have access from the outside, declared in one of the three levels `ShareLevel` (`Private`, `Registered` and `Public`). You can also set an expiration date of public access. Software functionality is described using the command `Items Share`.

Name	Description
<u>GET Info</u>	Getting information about access parameters of specified item in the workspace of the logged-in user
<u>PUT Edit</u>	Changing information about access parameters of specified item in the workspace of the logged-in user.
<u>DELETE Reset</u>	Resetting access parameters of specified item in the workspace of the logged-in user to default. These are <code>ShareLevel</code> , established in <code>Private</code> (no public access), <code>ShareMode</code> set in <code>Download</code> , as well as the absence of <code>ShareExpires</code> (period of validity of link of public access).
<u>Items Attach (PUT)</u>	Attaching one item to another. Items of type <code>StiReportTemplateItem</code> and <code>StiFileItem</code> support attaching of elements. Keys of the attached elements are in the <code>AttachedItems</code> collection. Attaching of elements is necessary for data binding. For example, it is possible to attach <code>DataSource</code> and images (<code>FileItem</code>) to a <code>ReportTemplate</code> to use these data in the report. It is possible to attach the XSD-file to the XML-file to

	connect the scheme of the XML-document with its data.
Items Detach (PUT)	Detaching one item from another. Keys of the attached elements are in the AttachedItems collection.

1.2.7.8.1 GET Info

Description:

Getting information about access parameters of specified item in the workspace of the logged-in user

Url Structure:

`http://reports.stimulsoft.com/1/items/itemkey/share`

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The `itemkey` parameter in the URI is the key of item and indicates the item whose data on the parameters of public access you want to get.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: a690257860fe456aae4c852d41c12378" http:
```

Returns:

JSON-object containing the field with the description of the parameters of public access element. `ResultShareLevel` describes the level of public access (`Private`, `Registered` and `Public`). `ResultShareExpires` contains the expiration date of public access (can be omitted). `ResultUrl` contains a link for public access. The success of the command execution is checked against the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "ItemGetShareInfo",
  "ResultShareLevel": "Private",
  "ResultShareExpires": "\/Date(1427801160000+0300)\/",
```

```
"ResultUrl": "http://reports.stimulsoft.com/s/1",  
"ResultSuccess": true  
}  
...
```

1.2.7.8.2 PUT Edit

Description:

Changing information about access parameters of specified item in the workspace of the logged-in user.

Url Structure:

`http://reports.stimulsoft.com/1/items/itemkey/share`

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The itemkey parameter in the URI is the key of item and indicates the item whose data of the parameters of public access you want to change. In POST-data must specify the JSON-object, which describing the changes in sharing:

POST-data in the JSON-object

```
...  
{  
  'ShareLevel': 'Public'  
}  
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: a690257860fe456aae4c852d41c12378" -d "{
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...  
{  
  "Ident": "ItemSetShareInfo",
```

```
"ResultSuccess": true
}
...
```

1.2.7.8.3 DELETE Reset

Description:

Resetting access parameters of specified item in the workspace of the logged-in user to default. These are `ShareLevel`, established in `Private` (no public access), `ShareMode` set in `Download`, as well as the absence of `ShareExpires` (period of validity of link of public access).

Url Structure:

`http://reports.stimulsoft.com/1/items/itemkey/share`

Method:

DELETE

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The itemkey parameter in the URI is the key of item and indicates the item whose data of the parameters of public access you want to set to default.

CURL example:

```
curl -X DELETE -H "x-sti-SessionKey: a690257860fe456aae4c852d41c12378" ht
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...
{
  "Ident": "ItemSetShareInfo",
  "ResultSuccess": true
}
...
```

1.2.7.8.4 Items Attach

Description:

Attaching one item to another. Items of type `StiReportTemplateItem` and

`StiFileItem` support attaching of elements. Keys of the attached elements are in the `AttachedItems` collection. Attaching of elements is necessary for data binding. For example, it is possible to attach `DataSource` and images (`FileItem`) to a `ReportTemplate` to use these data in the report. It is possible to attach the XSD-file to the XML-file to connect the scheme of the XML-document with its data.

Url Structure:

`http://reports.stimulsoft.com/1/items/itemkey/attach`

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-DestinationItemKey` contains the attached element. The `itemkey` parameter in the URI is the key of item and indicates the element to which must be attached an element.

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: da5053abac4544e9856e05bbda14f46a" -H "x
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...
{
  "Ident": "ItemSave",
  "ResultSuccess": true
}
...
```

1.2.7.8.5 Items Detach

Description:

Detaching one item from another. Keys of the attached elements are in the `AttachedItems` collection.

Url Structure:

`http://reports.stimulsoft.com/1/items/itemkey/detach`

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.
A custom header `x-sti-DestinationItemKey` contains the detached element.
The itemkey parameter in the URI is the key of item and indicates the element to which must be attached an element.

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: da5053abac4544e9856e05bbda14f46a" -H "x-
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...  
{  
  "Ident": "ItemSave",  
  "ResultSuccess": true  
}  
...
```

1.2.8 Scheduler

Schedulers object describes schedulers - special data elements designed to run different tasks on schedule. This provides the possibility of automation of complex data processing. To get the list of schedulers, information about the status, and launch new tasks in the current workspace, use the command **Schedulers** with different methods. Since schedulers are one type of item, then getting a list of schedulers and detailed information about the scheduler, as well as creating, modifying, and deleting of schedulers may to produce the same as any other items (use the **Items** command). However, the scheduler have several unique action, so to work with schedulers use the following command.

Name	Description
GET List	Getting a list of schedulers in a workspace of the logged-in user. The

	list is returned to the specified folder.
GET Info	Getting information about the scheduler in a workspace of the logged-in user.
POST Create	Creating a new scheduler in a workspace of the logged-in user. To successfully run the command you must fill the field Scheduler, which is an embedded object and describes the frequency of execution of actions, and contain a collection Actions, which describes a chain of actions. FolderKey field may contain the key of the parent folder, making sure that the scheduler in the tree. If FolderKey empty or not specified, the scheduler is displayed in the root folder.
PUT Edit	Changing scheduler in a workspace of the logged-in user. This command does not allow change of the unique scheduler key, which is used as an identifier (field Key), and the type (field Ident = 'SchedulerItem'). Changing field FolderKey can move a scheduler to another folder in the tree (the root, if you specify a null value).
DELETE	Removing a scheduler from the current workspace. However, the use of this command does not guarantee the immediate removal of the scheduler from a tree, because the command only creates an internal task of server to delete the scheduler and the actual deletion may be delayed for some time.

The scheduler can have two states - running (track events and time, and depending on this, the scheduler can run) and stopped (not tracked any external state). The state of the scheduler described by two values - Started or Stopped.

[GET Info \(Status\)](#)

Getting status of the scheduler.

[PUT Edit \(Status\)](#)

Setting status of the scheduler.

The actions described in the scheduler are performed according to a schedule or event. To run the command immediately, use the command Run.

[Run](#)

Manual start of actions the scheduler.

1.2.8.1 GET List

Description:

Getting a list of schedulers in a workspace of the logged-in user. The list is returned to the specified folder.

Url Structure:

`http://reports.stimulsoft.com/1/schedulers`

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. Custom header `x-sti-ItemKey` used to identify the parent folder, which list of schedulers needs to be retrieved. If this header is not present, it will get a list of schedulers of the root folder.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: ed46247f12fc44cf92f284ff5c8ffc12" http:
```

Returns:

The JSON object containing the collection `ResultItems`, which contains a list of schedulers in the specified folder of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample **JSON** response

```
...
{
  "Ident": "ItemFetchAll",
  "ResultItems": [
    {
      "Ident": "SchedulerItem",
      "Scheduler": {
        "Ident": "Hourly",
        "RunAtMinute": 50,
        "Status": "Started",
        "TimeZone": "Belarus Standard Time",
        "NotifyUsers": [],
        "LastTime": "\\Date(1425290450582)\\",
        "NextTime": "\\Date(1425293400000)\\",
        "Actions": [
          {
            "Ident": "RunReportAction",
            "ReportTemplateItemKey": "335f165d59f6426ba427477dc44c6758",
            "ResultType": "ReportSnapshot",
            "FileName": "",
          }
        ]
      },
      "StateKey": "1",
      "WorkspaceKey": "3c153c20e49f46219575ea4f9074888e",
      "Name": "HourlyScheduler",
      "Description": "",
      "Created": "\\Date(1425047855537)\\",
      "Modified": "\\Date(1425047855537)\\",
      "IsMoveable": true,
      "Key": "871a9f8275214f4cbc1a563c9ce28e5c"
    }
  ],
  "ResultSuccess": true
}
...
```

1.2.8.2 GET Info**Description:**

Getting information about the scheduler in a workspace of the logged-in user.

Url Structure:

<http://reports.stimulsoft.com/1/schedulers/schedulerkey>

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The `schedulerkey` parameter in the URI is the key of the scheduler and indicates the scheduler whose data you want to get.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: ea8cc765d54241e18347a043e187ada3" http:
```

Returns:

The JSON object containing the field `ResultItem`, which is the required scheduler of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "ItemGet",
  "ResultItem": {
    "Ident": "SchedulerItem",
    "Scheduler": {
      "Ident": "Hourly",
      "RunAtMinute": 50,
      "Status": "Started",
      "TimeZone": "Greenwich Standard Time",
      "NotifyUsers": [],
      "NextTime": "\\Date(1425300600000)\\",
      "Actions": [
        {
          "Ident": "RunReportAction",
          "ReportTemplateItemKey": "c68ec75e239b4d1a8f2032fd2d204629",
          "ResultType": "ReportSnapshot",
          "FileName": "",
          "AttachmentDelivery": "Link"
        }
      ]
    },
    "StateKey": "2",
    "WorkspaceKey": "3c153c20e49f46219575ea4f9074888e",
    "Name": "HourlyScheduler",
    "Description": "Edited Description",
    "Created": "\\Date(1425295716657)\\",
    "Modified": "\\Date(1425299145087)\\",
    "IsMoveable": true,
    "Key": "6a447a392c454325ba436629b827644b"
  },
  "ResultSuccess": true
}
...
```

1.2.8.3 POST Create

Description:

Creating a new scheduler in a workspace of the logged-in user. To successfully run the command you must fill the field Scheduler, which is an embedded object and describes the frequency of execution of actions, and contain a collection Actions, which describes a chain of actions. FolderKey field may contain the key of the parent folder, making sure that the scheduler in the tree. If FolderKey empty or not specified, the scheduler is displayed in the root folder.

Url Structure:

`http://reports.stimulsoft.com/1/schedulers`

Method:

POST

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. In POST-data must specify the JSON-object, which describing the new scheduler:

POST-data in the JSON-object

```
...
{
  'Ident': 'SchedulerItem',
  'Scheduler':
  {
    'Ident': 'Hourly',
    'RunAtMinute': 30,
    'Status': 'Started',
    'TimeZone': 'Greenwich Standard Time',
    'NotifyUsers': [
    ],
    'Actions': [
    {
      'Ident': 'RunReportAction',
      'ReportTemplateItemKey': 'c68ec75e239b4d1a8f2032fd2d204629',
      'ResultType': 'ReportSnapshot',
      'FileItemName': '',
      'AttachmentDelivery': 'Link'
    }
    ]
  },
  'Name': 'NewHourlyScheduler',
  'Description': 'Created Hourly Scheduler'
}
...
```

CURL example:

```
curl-XPOST-H"x-sti-SessionKey: ed46247f12fc44cf92f284ff5c8ffc12"- d '{"Id': 'ItemSave', 'NotifyUsers': [], 'Actions': [{ 'Ident': 'RunReportAction', 'ReportTemplateItem': 'ReportTemplateItem' } ]}' http://reports.stimulsoft.com/1/schedulers
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...
{
  "Ident": "ItemSave",
  "ResultSuccess": true
}
...
```

1.2.8.4 PUT Edit**Description:**

Changing scheduler in a workspace of the logged-in user. This command does not allow change of the unique scheduler key, which is used as an identifier (field `Key`), and the type (field `Ident` = 'SchedulerItem'). Changing field `FolderKey` can move a scheduler to another folder in the tree (the root, if you specify a null value).

Url Structure:

```
http://reports.stimulsoft.com/1/schedulers/schedulerkey
```

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The `schedulerkey` parameter in the URI is the key of scheduler and indicates the scheduler whose data you want to change. In POST-data must specify the JSON-object describing the changed scheduler data:

POST-data in the JSON-object

```
...
{
  'Description': 'Edited Description'
}
```

```
}  
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ed46247f12fc44cf92f284ff5c8ffc12" -d "{
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...  
{  
  "Ident": "ItemSave",  
  "ResultSuccess": true  
}  
...
```

1.2.8.5 DELETE**Description:**

Removing a scheduler from the current workspace. However, the use of this command does not guarantee the immediate removal of the scheduler from a tree, because the command only creates an internal task of server to delete the scheduler and the actual deletion may be delayed for some time.

Url Structure:

`http://reports.stimulsoft.com/1/schedulers/schedulerkey`

Method:

`DELETE`

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-AllowMoveToRecycleBin` allows deleting an item to the recycle bin, by default it is set to `true`. To remove an item with referenced resources set `x-sti-AllowMoveToRecycleBin` to `false`. The `schedulerkey` parameter in the URI is the key of scheduler and indicates the scheduler whose data you want to delete.

CURL example:

```
curl -X DELETE -H "x-sti-SessionKey: ed46247f12fc44cf92f284ff5c8ffc12" -H
```

Returns:

The success of the command execution is checked by the content of the field `ResultSuccess`. `ResultTaskKey` field contains unique key of internal server tasks, created to remove the scheduler.

Sample JSON response

```
...
{
  "Ident": "ItemDelete",
  "ResultTaskKey": "775713fd9614415b9578bcf60c65f7c9",
  "ResultSuccess": true
}
...
```

1.2.8.6 GET_Info (Status)**Description:**

Getting status of the scheduler.

Url Structure:

```
http://reports.stimulsoft.com/1/schedulers/schedulerkey/status
```

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The `schedulerkey` parameter in the URI is the key of scheduler and indicates the scheduler whose state you want to get.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: ed46247f12fc44cf92f284ff5c8ffc12" http:
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully. `ResultStatus` field contains the current state of the scheduler.

Sample JSON response

```
...
{
  "Ident": "SchedulerGetStatus",
  "ResultStatus": "Started",
  "ResultSuccess": true
}
...
```

1.2.8.7 PUT Edit (Status)

Description:

Setting status of the scheduler.

Url Structure:

`http://reports.stimulsoft.com/1/schedulers/schedulerkey/status`

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.
A custom header `x-sti-Status` contains the required state of scheduler (Started or Stopped). The `schedulerkey` parameter in the URI is the key of scheduler and indicates the scheduler whose state you want to set.

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ed46247f12fc44cf92f284ff5c8ffc12" -H "x
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...
{
  "Ident": "SchedulerSetStatus",
  "ResultSuccess": true
}
...
```

1.2.8.8 Run

The actions described in the scheduler are performed according to a schedule or event. To run the command immediately, use the command `Run`.

Description:

Manual start of actions the scheduler.

Url Structure:

`http://reports.stimulsoft.com/1/schedulers/schedulerkey/run`

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The `schedulerkey` parameter in the URI is the key of scheduler and indicates the scheduler whose you want to run.

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ed46247f12fc44cf92f284ff5c8ffc12" -d ""
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully. `ResultTaskKey` field contains unique key of internal server tasks, set up to run the scheduler's actions.

Sample JSON response

```
...
{
  "Ident": "SchedulerRun",
  "ResultTaskKey": "412d7812d60d449db380ef3ccb51e80d",
  "ResultSuccess": true
}
...
```

1.2.9 Files

Files object describes files - special data elements designed to management of external data and uploading/downloading of user files used to generate reports (data to be imported, images, etc.). This provides the possibility of external resources processing. `FileType` property of `FileItem` describes the content type of the file

and can have values from this table:

Identifier	Description
Unknown	Unsupported file type
ReportSnapshot	Rendered report
Pdf	PDF-file
Xps	XPS-file
Html	HTML-file
Text	Text file
RichText	RichText file format (RTF)
Word	MS Word document file
Excel	MS Excel document file
PowerPoint	MS PowerPoint presentation file
OpenDocumentWriter	OpenDocument-file for Writer
OpenDocumentCalc	OpenDocument-file for Calc
Data	One of multiple data format
Image	One of multiple image format
Xml	XML-file
Xsd	XSD-file
Csv	CSV-file
Dbf	DBF-file
Sylk	Sylk-file
Dif	Dif-file
Json	Data in JSON format

To get the list of files, download and upload resources use the command Files with different methods. Since files are one type of item, then getting a list of files and detailed information about the files, modifying and deleting of files may to produce the same as any other items (use the Items command). However, the files have several unique action, so to work with files use the following command.

Name	Description
GET List	Getting a list of files in a workspace of the logged-in user. The list is returned to the specified folder.
GET Download	Getting information about the file (including Base64-encoded resource) in a workspace of the logged-in user.
POST Create	Creating a new file in a workspace of the logged-in user. This command creates an element of type <code>StiFileItem</code> and allows you to load the resource file (no more than 90Kb) encoded in base64. If the resource file size exceeds 90Kb, it is necessary to split the file into chunks (not exceeding 90 Kb), encode them to <code>Base64</code> and send first chunk with this command, and all the other chunks upload using the command described below. <code>FolderKey</code> field may contain the key of the parent folder, making sure that the file in the tree. If <code>FolderKey</code> empty or not specified, the file is displayed in the root folder.
PUT Append	Upload a new chunk of file in a workspace of the logged-in user. Chunk size must be less than 90Kb and encoded in Base64-form.
DELETE	Removing a file from the current

workspace. However, the use of this command does not guarantee the immediate removal of the file from a tree, because the command only creates an internal task of server to delete the file and the actual deletion may be delayed for some time.

This command remove file item and resources without using the recycle bin.

1.2.9.1 GET List

Description:

Getting a list of files in a workspace of the logged-in user. The list is returned to the specified folder.

Url Structure:

`http://reports.stimulsoft.com/1/files`

Method:

`GET`

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. Custom header `x-sti-ItemKey` used to identify the parent folder, which list of files needs to be retrieved. If this header is not present, it will get a list of files of the root folder.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: 2aa74b48f7c542b9a17cbcfa5d43122d" http:
```

Returns:

The JSON object containing the collection `ResultItems`, which contains a list of files in the specified folder of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...  
{
```

```
"Ident": "ItemFetchAll",
"ResultItems": [
{
  "Ident": "FileItem",
  "FileType": "Csv",
  "WorkspaceKey": "1b11a087888f4a968ecbbaf74423647c",
  "Name": "Small.csv",
  "Description": "",
  "Created": "\\Date(1427458480000)\\",
  "Modified": "\\Date(1427458480000)\\",
  "IsMoveable": true,
  "Key": "53efd43b18b7455e88d6013369473772"
},
{
  "Ident": "FileItem",
  "FileType": "Excel",
  "WorkspaceKey": "1b11a087888f4a968ecbbaf74423647c",
  "Name": "stat.xlsx",
  "Description": "",
  "Created": "\\Date(1427455740000)\\",
  "Modified": "\\Date(1427455740000)\\",
  "IsMoveable": true,
  "Key": "387df1f56670479897cd81716358f626"
}
],
"ResultSuccess": true
}
...
```

1.2.9.2 GET Download

Description:

Getting information about the file (including Base64-encoded resource) in a workspace of the logged-in user.

Url Structure:

`http://reports.stimulsoft.com/1/files/filekey`

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The filekey parameter in the URI is the key of the file item and indicates the file whose data you want to get.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: 2aa74b48f7c542b9a17cbcfa5d43122d" http:
```

Returns:

The JSON object containing the collection `ResultCommands` with two items. The first has `Ident` `ItemGet` and describes the element `StiFileItem`, the second has `Ident` `ItemResourceGet` and describes the resource (file contents).

`ResultResource` field contains data from a file in Base64-form. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "CommandListRun",
  "ResultCommands": [
    {
      "Ident": "ItemGet",
      "ItemKey": "53efd43b18b7455e88d6013369473772",
      "ResultItem":
      {
        "Ident": "FileItem",
        "FileType": "Csv",
        "WorkspaceKey": "1b11a087888f4a968ecbbaf74423647c",
        "Name": "Small.csv",
        "Description": "",
        "Created": "\\Date(1427458480000)\\",
        "Modified": "\\Date(1427458480000)\\",
        "IsMoveable": true,
        "Key": "53efd43b18b7455e88d6013369473772"
      },
      "ResultLastVersionKey": "3af1a24040fe4fe18916376d405720d2",
      "SessionKey": "2aa74b48f7c542b9a17cbcfa5d43122d",
      "ResultSuccess": true
    },
    {
      "Ident": "ItemResourceGet",
      "ItemKey": "53efd43b18b7455e88d6013369473772",
      "ResultResource":
      "aWQ7UGVyc29u01R5cGU7QVNFDQox03Rlc3RfMDE7TTsxzDQoy03Rlc3RfMDI7RjszDQoz03Rlc3RfMDM7Rjs0DQo003Rlc3RfMDQ7TTs0DQo103Rlc3RfMDU7RjsxDQo203Rlc3RfMDY7TTszDQo303Rlc3RfMDC7TTsyDQo403Rlc3RfMDg7Rjs0DQo503Rlc3RfMDk7RjsxDQoxMDt0ZXN0XzEw0007NA0KMTc7dGVzdF8xMTtNOzQNCjEy03Rlc3RfMTI7TTszDQoxMzt0ZXN0XzEz00Y7MQ0KMTQ7dGVzdF8xNDtGOzQNCjE103Rlc3RfMTU7RjsyDQoxNjt0ZXN0XzE200Y7Mw0KMTc7dGVzdF8xNztGOzENCjE403Rlc3RfMTg7TTs0DQoxOTt0ZXN0XzE50007MQ0KMjA7dGVzdF8yMDtGOzINCg==",
      "SessionKey": "2aa74b48f7c542b9a17cbcfa5d43122d",
      "ResultSuccess": true
    }
  ],
  "ResultSuccess": true
}
...
```

1.2.9.3 POST Create

Description:

Creating a new file in a workspace of the logged-in user. This command creates an element of type `StiFileItem` and allows you to load the resource file (no more than 90Kb) encoded in `base64`. If the resource file size exceeds 90Kb, it is necessary to split the file into chunks (not exceeding 90 Kb), encode them to `Base64` and send first chunk with this command, and all the other chunks upload using the command described below. `FolderKey` field may contain the key of the parent folder, making sure that the file in the tree. If `FolderKey` empty or not specified, the file is displayed in the root folder.

Url Structure:

`http://reports.stimulsoft.com/1/files`

Method:

`POST`

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. In POST-data must specify the JSON-object, which describing the new file with `Base64`-encoded resource data:

POST-data in the JSON-object

```
...
{
  'Ident': 'FileItem',
  'Name': 'Test',
  'Description': 'This is a TEST',
  'FileType': 'Excel',
  'Resource': 'dGVzdCBzdHJpbmcgZGF0YQ=='
}
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: 2aa74b48f7c542b9a17cbcfa5d43122d" -d "
```

Returns:

The JSON object containing the collection `ResultCommands` with two items. The first has Ident `ItemSave` and describes the created element `StiFileItem`. The field

Key contain the key of created file item. The second element of root collection has Ident `ItemResourceSave` and contain the field `ResultVersionKey`. The value of this field (version key) need to download pieces of the file. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample **JSON** response

```
...
{
  "Ident": "CommandListRun",
  "ResultCommands": [
    {
      "Ident": "ItemSave",
      "Items": [
        {
          "Ident": "FileItem",
          "FileType": "Excel",
          "WorkspaceKey": "1b11a087888f4a968ecbbaf74423647c",
          "Name": "Test",
          "Description": "This is a TEST",
          "Created": "\\Date(1427464242386)\\",
          "Modified": "\\Date(1427464242386)\\",
          "IsMoveable": true,
          "Key": "bad8820fd26e4ab7b04dcc5afa932148"
        }
      ],
      "SessionKey": "2aa74b48f7c542b9a17cbcfa5d43122d",
      "ResultSuccess": true
    },
    {
      "Ident": "ItemResourceSave",
      "ResultVersionKey": "74cc142d844a496797d78b7e34b49687",
      "ResultSuccess": true
    }
  ],
  "ResultSuccess": true
}
...
```

1.2.9.4 PUT Append

Description:

Upload a new chunk of file in a workspace of the logged-in user. Chunk size must be less than 90Kb and encoded in `Base64`-form.

Url Structure:

`http://reports.stimulsoft.com/1/files/filekey`

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-VersionKey` contains the version key of previous chunk (obtained in the previous step). The `filekey` parameter in the URI is the key of file item and indicates the file whose data you want to upload. In POST-data must specify the JSON-object describing the resources data with only one field. Data must be Base64-encoded:

POST-data in the JSON-object

```
...
{
  'Resource': 'IGFwcGVuZGVkIHNOcm1uZw=='
}
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: 2aa74b48f7c542b9a17cbcf5d43122d" -H "x
```

Returns:

The JSON object containing the collection `ResultCommands` which contain once item with field `ResultVersionKey`. Value of this field need to upload next chunk. Data will be added to the end of file.

Sample JSON response

```
...
{
  "Ident": "CommandListRun",
  "ResultCommands": [
    {
      "Ident": "ItemResourceSave",
      "ResultVersionKey": "74cc142d844a496797d78b7e34b49687",
      "ResultSuccess": true
    }
  ],
  "ResultSuccess": true
}
...
```

1.2.9.5 DELETE

Description:

Removing a file from the current workspace. However, the use of this command does not guarantee the immediate removal of the file from a tree, because the command only creates an internal task of server to delete the file and the actual deletion may be delayed for some time. This command remove file item and resources without using the recycle bin.

Url Structure:

`http://reports.stimulsoft.com/1/files/filekey`

Method:

`DELETE`

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The `filekey` parameter in the URI is the key of file and indicates the file whose data and resources you want to delete.

CURL example:

```
curl -X DELETE -H "x-sti-SessionKey: 2aa74b48f7c542b9a17cbcfa5d43122d" ht
```

Returns:

The success of the command execution is checked by the content of the field `ResultSuccess`. `ResultTaskKey` field contains unique key of internal server tasks, created to remove the file item.

Sample JSON response

```
...
{
  "Ident": "ItemDelete",
  "ResultTaskKey": "8f70f36181c148c287c3a6721d9d3283",
  "ResultSuccess": true
}
...
```

1.2.10 Report Template

`ReportTemplate` object describes report templates - special data elements what contains the structure of the report. To build this report you need to execute command `Run`. As a result was obtained an element of type `StiReportSnapshot` - rendered report with data. Since report templates are one type of item, then getting

a list of report templates and detailed information about the report templates, as well as creating and deleting of report templates may to produce the same as any other items (use the Items command). However, the report templates have several unique action, so to work with report templates use the following command.

Name	Description
GET List	Getting a list of report templates in a workspace of the logged-in user. The list is returned to the specified folder.
GET Info	Getting information about the report templates in a workspace of the logged-in user.
POST Create	Creating a new report template in a workspace of the logged-in user. <code>FolderKey</code> field may contain the key of the parent folder, making sure that the report template in the tree. If <code>FolderKey</code> empty or not specified, the report template is displayed in the root folder.
DELETE	Removing a report template from the current workspace. However, the use of this command does not guarantee the immediate removal of the report template from a tree, because the command only creates an internal task of server to delete the report template and the actual deletion may be delayed for some time.

The command is used to render report. To work correctly, you must first create two items – a report template what will be run, and the destination item where data will be stored. The source item is the current `StiReportTemplate` object. The destination item is the object of type [StiFileItem](#) or [StiReportSnapshotItem](#). If the destination item is the `StiFileItem`, then report is converted to the format specified when creating file item (field 'FileType'). If the destination item is the

StiReportSnapshotItem, then rendered report will be saved in the report snapshot item.

[Run](#)

Manual starting of actions of the scheduler.

[POST Duplicate](#)

Creating a new copy of the report template in a workspace of the logged-in user.

1.2.10.1 GET List

Description:

Getting a list of report templates in a workspace of the logged-in user. The list is returned to the specified folder.

Url Structure:

`http://reports.stimulsoft.com/1/reporttemplates`

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. Custom header `x-sti-ItemKey` used to identify the parent folder, which list of report templates needs to be retrieved. If this header is not present, it will get a list of report templates of the root folder.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: 1add6a4f1c5e481c80e964b613ee6089" http:
```

Returns:

The JSON object containing the collection `ResultItems`, which contains a list of report templates in the specified folder of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "ItemFetchAll",
  "ResultItems": [
    {
      "Ident": "ReportTemplateItem",
```

```
"StateKey": "e",
"AttachedItems": [
  "53efd43b18b7455e88d6013369473772"
],
"WorkspaceKey": "1b11a087888f4a968ecbbaf74423647c",
"Name": "Report",
"Description": "",
"Created": "\\Date(1427718722000)\\",
"Modified": "\\Date(1427795741000)\\",
"IsMoveable": true,
"Key": "83a5f6d43351499fa9a2d40822f5772b"
}
],
"ResultSuccess": true
}
...
```

1.2.10.2 GET Info

Description:

Getting information about the report templates in a workspace of the logged-in user.

Url Structure:

<http://reports.stimulsoft.com/1/reporttemplates/reporttemplatekey>

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The `reporttemplatekey` parameter in the URI is the key of the report templates and indicates the report templates whose data you want to get.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: 1add6a4f1c5e481c80e964b613ee6089" http:
```

Returns:

The JSON object containing the field `ResultItem`, which is the required report template of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

...

```
{
  "Ident": "ItemGet",
  "ResultItem":
  {
    "Ident": "ReportTemplateItem",
    "StateKey": "e",
    "AttachedItems": [
      "53efd43b18b7455e88d6013369473772"
    ],
    "WorkspaceKey": "1b11a087888f4a968ecbbaf74423647c",
    "Name": "Report",
    "Description": "",
    "Created": "\\Date(1427718722000)\\",
    "Modified": "\\Date(1427795741000)\\",
    "IsMoveable": true,
    "Key": "83a5f6d43351499fa9a2d40822f5772b"
  },
  "ResultLastVersionKey": "4ae62efae4d74b6aab74567980e36b19",
  "ResultSuccess": true
}
...
```

1.2.10.3 POST Create

Description:

Creating a new report template in a workspace of the logged-in user. FolderKey field may contain the key of the parent folder, making sure that the report template in the tree. If FolderKey empty or not specified, the report template is displayed in the root folder.

Url Structure:

`http://reports.stimulsoft.com/1/reporttemplates`

Method:

POST

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. In POST-data must specify the JSON-object, which describing the new report template:

POST-data in the JSON-object

```
...
{
  'Ident': 'ReportTemplateItem',
  'Name': 'NewReportTemplate',
  'Description': 'This is a new report template'
```

```
}  
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: 1add6a4f1c5e481c80e964b613ee6089" -d "
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...  
{  
  "Ident": "ItemSave",  
  "ResultSuccess": true  
}  
...
```

1.2.10.4 DELETE**Description:**

Removing a report template from the current workspace. However, the use of this command does not guarantee the immediate removal of the report template from a tree, because the command only creates an internal task of server to delete the report template and the actual deletion may be delayed for some time.

Url Structure:

`http://reports.stimulsoft.com/1/reporttemplates/reporttemplatekey`

Method:

`DELETE`

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-AllowMoveToRecycleBin` allows deleting an item to the recycle bin, by default it is set to `true`. To remove an item with referenced resources set `x-sti-AllowMoveToRecycleBin` to `false`. The `reporttemplatekey` parameter in the URI is the key of report template and indicates the report template whose data you want to delete.

CURL example:

```
curl -X DELETE -H "x-sti-SessionKey: 1add6a4f1c5e481c80e964b613ee6089" -H
```

Returns:

The success of the command execution is checked by the content of the field `ResultSuccess`. `ResultTaskKey` field contains unique key of internal server tasks, created to remove the report template.

Sample JSON response

```
...
{
  "Ident": "ItemDelete",
  "ResultTaskKey": "2728300b164f4a358c6df65ee7ab9304",
  "ResultSuccess": true
}
...
```

1.2.10.5 Run

The command is used to render report. To work correctly, you must first create two items – a report template what will be run, and the destination item where data will be stored. The source item is the current `StiReportTemplate` object. The destination item is the object of type `StiFileItem` or `StiReportSnapshotItem`. If the destination item is the `StiFileItem`, then report is converted to the format specified when creating file item (field `'FileType'`). If the destination item is the `StiReportSnapshotItem`, then rendered report will be saved in the report snapshot item.

Description:

Manual starting of actions of the scheduler.

Url Structure:

`http://reports.stimulsoft.com/1/reporttemplates/reporttemplatekey/run`

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-DestinationItemKey` contains the destination item key (an item with ident `FileItem` and `ReportSnapshotItem`). The `reporttemplatekey` parameter in the URI is the key of report template and indicates the report template whose you want to run.

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: 1add6a4f1c5e481c80e964b613ee6089" -H "x
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully. `ResultTaskKey` field contains unique key of internal server tasks, set up to run the rendering of report template.

Sample JSON response

```
...
{
  "Ident": "ReportRun",
  "ResultTaskKey": "c375e1f4cd5645bab669bd124488329d",
  "ResultSuccess": true
}
...
```

To run the report with parameters, use the JSON data object as in the example below:

```
curl -X PUT -d '{"Parameters': [{ 'Ident': 'Single', 'Name': 'param', 'Type'
```

In the POST data you must specify the JSON object that describes the parameters of the report template:

POST-data in the JSON-object

```
...
{
  'Parameters': [
    {
      'Ident': 'Single',
      'Name': 'param',
      'Type': 'string',
      'Value': 'testvalue'
    }
  ]
}
...
```

Ident parameter may have values - 'Single', 'Range' and 'List'.

Sample JSON response

```

...
{
  "Ident": "ReportRun",
  "Parameters": [
    {
      "Ident": "Single",
      "Name": "param",
      "Type": "string",
      "Value": "testvalue"
    },
    {
      "Ident": "Range",
      "Name": "rangeParamName",
      "FromValue": "1",
      "ToValue": "2",
      "RangeType": "StringRange",
      "FromType": "string",
      "ToType": "string"
    },
    {
      "Ident": "List",
      "Name": "listParamName",
      "ListType": "StringList",
      "Values": [{
        "Ident": "Single",
        "Value": "11",
        "Type": "string"
      },
      {
        "Ident": "Single",
        "Value": "12",
        "Type": "string"
      },
      {
        "Ident": "Single",
        "Value": "13",
        "Type": "string"
      }
    ]
  },
  "ReportTemplateItemKey": "55617f45d7c641dc9b806f6859c995d1",
  "UserKey": "f215f398a01740638ef02e8e9437b40c",
  "TaskKey": "806019df4aec44elaeeeb09a169e20f7",
  "ResultTaskKey": "806019df4aec44elaeeeb09a169e20f7",
  "SessionKey": "8a8d4554fb564422a7232107a8ab29f3",
  "ResultSuccess": true
}
...

```

1.2.10.6 POST Duplicate

Description:

Creating a new copy of the report template in a cloud of the logged-in user.

Url Structure:

```
https://cloud.stimulsoft.com/1/reporttemplates
```

Method:

POST

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The `reporttemplatekey` parameter in the URI is the key of report template and indicates the report template whose data you want to copy.

CURL example:

```
curl -X POST -H "x-sti-SessionKey: 3fd143cd876048a188a6a3d69da0f535" -d "
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...
{
  "Ident": "CommandListRun",
  "ContinueAfterError": false,
  "ResultCommands": [
    {
      "Ident": "ItemSave",
      "AllowSignalsReturn": false,
      "SaveEmptyResources": false,
      "ResultItems": [
        {
          "Ident": "ReportTemplateItem",
          "RefreshFrequency": "Always",
          "CacheMode": "Off",
          "StateKey": "1",
          "ShareLevel": "Private",
          "WorkspaceKey": "b150683855854affbc98b142d4c61cea",
          "Name": "ChartStyle_Copy",
          "Created": "\\Date(1644323381839)\\",
          "Modified": "\\Date(1644323381839)\\",
          "Visible": true,
          "Deleted": false,
          "IsFolder": false,
          "IsMoveable": true,
          "Key": "bb4af6d48c304d558060334988af1291"
        }
      ]
    },
    {
      "Ident": "ItemResourceSave",
```

```

        "Type": "Insert",
        "ResultVersionKey": "789221129c624a25b2a46f553c0c66d6",
        "ResultSuccess": true
    }
],
"ResultSuccess": true
}
...

```

1.2.11 Report Snapshot

`ReportSnapshot` object describes report snapshots - special data elements what contains the rendered report. Since `ReportSnapshot` are one type of item, then getting a list of report snapshots and detailed information about the report snapshots, as well as creating and deleting of report snapshots may to produce the same as any other items (use the `Items` command). However, the report snapshots have several unique action, so to work with report snapshots use the following command.

Name	Description
GET List	Getting a list of report snapshots in a workspace of the logged-in user. The list is returned to the specified folder.
GET Info	Getting information about the report snapshots in a workspace of the logged-in user.
POST Create	Creating a new report snapshot in a workspace of the logged-in user. <code>FolderKey</code> field may contain the key of the parent folder, making sure that the report snapshot in the tree. If <code>FolderKey</code> empty or not specified, the report snapshot is displayed in the root folder.
DELETE	Removing a report snapshot from the current workspace. However, the use of this command does not guarantee the immediate removal of the report snapshot from a tree, because the command only creates

an internal task of server to delete the report snapshot and the actual deletion may be delayed for some time.

The command is used to export data from report snapshot. Data from a snapshot can be exported to one of a supported formats by one of two ways:

- You must create a `FileItem` of specific type, which you want to export data. Then run the export command specifying the key of the `FileItem` in the custom header `x-sti-DestinationItemKey`;
- You can export data directly, specifying in the POST-data a JSON-object containing object `ExportSet` - a list of options to export, individual for each data type.

[Export](#)

Export data from a `ReportSnapshot` to a `FileItem`.

1.2.11.1 GET List

Description:

Getting a list of report snapshots in a workspace of the logged-in user. The list is returned to the specified folder.

Url Structure:

`http://reports.stimulsoft.com/1/reportsnapshots`

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. Custom header `x-sti-ItemKey` used to identify the parent folder, which list of report snapshots needs to be retrieved. If this header is not present, it will get a list of report snapshots of the root folder.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: aea7a6197b8c481eaf6077178f69f2bd" http:
```

Returns:

The JSON object containing the collection `ResultItems`, which contains a list of report snapshots in the specified folder of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "ItemFetchAll",
  "ResultItems": [
    {
      "Ident": "ReportSnapshotItem",
      "WorkspaceKey": "1b11a087888f4a968ecbbaf74423647c",
      "Name": "ReportSnapshot",
      "Created": "\\Date(1428933145000)\\",
      "Modified": "\\Date(1428933145000)\\",
      "IsMoveable": true,
      "Key": "4625a71ef47d46c3b5db7b5185a89e5b"
    },
    {
      "Ident": "ReportSnapshotItem",
      "WorkspaceKey": "1b11a087888f4a968ecbbaf74423647c",
      "Name": "ReportSnapshot1",
      "Description": "",
      "Created": "\\Date(1428928320000)\\",
      "Modified": "\\Date(1428933138000)\\",
      "IsMoveable": true,
      "Key": "6289e011469c4ed492c061fb142ee983"
    }
  ],
  "ResultSuccess": true
}
...
```

1.2.11.2 GET Info

Description:

Getting information about the report snapshots in a workspace of the logged-in user.

Url Structure:

<http://reports.stimulsoft.com/1/reportsnapshots/reportsnapshotkey>

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.

The `reportsnapshotkey` parameter in the URI is the key of the report snapshots and indicates the report snapshots whose data you want to get.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: aea7a6197b8c481eaf6077178f69f2bd" http:
```

Returns:

The JSON object containing the field `ResultItem`, which is the required report snapshot of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "ItemGet",
  "ResultItem":
  {
    "Ident": "ReportSnapshotItem",
    "WorkspaceKey": "1b11a087888f4a968ecbbaf74423647c",
    "Name": "ReportSnapshot1",
    "Description": "",
    "Created": "\\Date(1428928320000)\\",
    "Modified": "\\Date(1428933138000)\\",
    "IsMoveable": true,
    "Key": "6289e011469c4ed492c061fb142ee983"
  },
  "ResultLastVersionKey": "3fce6a2f25c44ea2a23fce402054fb77",
  "ResultSuccess": true
}
...
```

1.2.11.3 POST Create**Description:**

Creating a new report snapshot in a workspace of the logged-in user. `FolderKey` field may contain the key of the parent folder, making sure that the report snapshot in the tree. If `FolderKey` empty or not specified, the report snapshot is displayed in the root folder.

Url Structure:

```
http://reports.stimulsoft.com/1/reportsnapshots
```

Method:

```
POST
```

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. In POST-data must specify the JSON-object, which describing the new report snapshot:

POST-data in the JSON-object

```
...
{
  'Ident': 'ReportSnapshotItem',
  'Name': 'NewReportSnapshot',
  'Description': 'This is a new report snapshot'
}
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: aea7a6197b8c481eaf6077178f69f2bd" -d "
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully.

Sample JSON response

```
...
{
  "Ident": "ItemSave",
  "ResultSuccess": true
}
...
```

1.2.11.4 DELETE**Description:**

Removing a report snapshot from the current workspace. However, the use of this command does not guarantee the immediate removal of the report snapshot from a tree, because the command only creates an internal task of server to delete the report snapshot and the actual deletion may be delayed for some time.

Url Structure:

`http://reports.stimulsoft.com/1/reportsnapshots/reportsnapshotkey`

Method:

DELETE

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-AllowMoveToRecycleBin` allows deleting an item to the recycle bin, by default it is set to `true`. To remove an item with referenced resources set `x-sti-AllowMoveToRecycleBin` to `false`. The `reportsnapshotkey` parameter in the URI is the key of report snapshot and indicates the report snapshot whose data you want to delete.

CURL example:

```
curl -X DELETE -H "x-sti-SessionKey: aea7a6197b8c481eaf6077178f69f2bd" -H
```

Returns:

The success of the command execution is checked by the content of the field `ResultSuccess`. `ResultTaskKey` field contains unique key of internal server tasks, created to remove the report snapshot.

Sample JSON response

```
...
{
  "Ident": "ItemDelete",
  "ResultTaskKey": "82d1fe4a5c0c4f0b9207daceb6a7ffa6",
  "ResultSuccess": true
}
...
```

1.2.11.5 Export

The command is used to export data from report snapshot. Data from a snapshot can be exported to one of a supported formats by one of two ways:

- You must create a `FileItem` of specific type, which you want to export data. Then run the export command specifying the key of the `FileItem` in the custom header `x-sti-DestinationItemKey`;
- You can export data directly, specifying in the POST-data a JSON-object containing object `ExportSet` - a list of options to export, individual for each data type.

Description:

Export data from a `ReportSnapshot` to a `FileItem`.

Url Structure:

`http://reports.stimulsoft.com/1/reportsnapshots/reportsnapshotkey/export`

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-DestinationItemKey` contains the destination item key (an item with ident "FileItem"). The `reportsnapshotkey` parameter in the URI is the key of report snapshot and indicates the report snapshot whose you want to export. In POST-data must specify the JSON-object, which describing the `ExportSet`-object (example for simple export to a PDF-file):

POST-data in the JSON-object

```
...
{
  'ExportSet':
  {
    'Ident': 'Pdf',
    'PageRange': { },
    'EmbeddedFonts': false,
    'DitheringType': 'None',
    'PdfACompliance': true
  }
}
...
```

More detail about `ExportSet`-object see the appendix.

CURL example:

1 way:

```
curl -X PUT -H "x-sti-SessionKey: aea7a6197b8c481eaf6077178f69f2bd" -H "x-
```

2 way:

```
curl -X PUT -H "x-sti-SessionKey: aea7a6197b8c481eaf6077178f69f2bd" -d "{
```

Returns:

The JSON object containing the field `ResultSuccess` which indicates that the command is executed successfully. `ResultTaskKey` field contains unique key of internal server tasks, set up to run the export data from report snapshot.

Sample JSON response

```
...
{
  "Ident": "ReportExport",
  "ResultTaskKey": "ba493223bc884ad19a6dab8b04b84c1d",
  "ResultSuccess": true
}
...
```

1.2.12 Export

`ExportSet`-object contain information about settings for data exporting to more files formats. Each of these formats supports unique features, so different types of files there are different sets of settings.

Full description of the options can be found in the documentation here https://www.stimulsoft.com/en/documentation/online/programming-manual/index.html?engine_exports.htm

Any `ExportSet`-object contain fields `'Ident'` and `'PageRange'`.

`Ident` describes the format in which you want to export data. Values for this field listed in the table below. `ExportSet` Idents:

Identifier	Description
PDF	PDF-file
XPS	XPS-file
PowerPoint	MS PowerPoint presentation file
HTML	HTML-file
Text	Text file
RichText	RichText file format (RTF)
Word	MS Word document file
OpenDocumentWriter	OpenDocument-file for Writer
Excel	MS Excel document file
OpenDocumentCalc	OpenDocument-file for Calc
Data	One of multiple data format
Image	One of multiple image format

`PageRange` describes the pages of the report, which need to be processed. There are three possible values: All pages, Current page and selected pages or range of pages (see the table below). `ExportSet` `PageRange`:

All	Current Page	Selected Pages (e.g. 1, 3, 5-7)
<code>"PageRange": {},</code>	<code>"PageRange": { "RangeType": 2 },</code>	<code>"PageRange": { "RangeType": 3, ,</code>

		" P a g e R a n g e s " : " 1 , 3 , 5 - 7 " },
--	--	--

1.2.12.1 PDF

There are two stages in the export of a report template:

- › [A file item is created for a future report;](#)
- › [Export a report template to the destination file item.](#)

Create Destination File Item

Url Structure:

`http://reports.stimulsoft.com/1/files`

Method:

POST

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.

A custom header `x-sti-SessionKey` contains the session key of the current user.

In POST-data must specify the JSON-object, which describing the new file item:

POST-data in the JSON-object

```
...
{
  'Ident': 'FormItem',
  'Name': 'PDF',
  'Description': 'This is a sample export to PDF',
  'FileType': 'Pdf'
}
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: 4ccbebaac38642b4a182d555aa09ee46" -d "
```

Returns:

The JSON object containing the field `ResultSessionKey`, which is a list of members of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "CommandListRun",
  "ContinueAfterError": false,
  "ResultCommands": [
    {
      "Ident": "ItemSave",
      "AllowSignalsReturn": false,
      "SaveEmptyResources": false,
      "ResultItems": [
        {
          "Ident": "FormItem",
          "FileType": "Pdf",
          "ShareLevel": "Private",
          "HasItems": false,
          "StateKey": "1",
          "WorkspaceKey": "8a87e146d96e4b2e9aa127b22d6d98df",
          "Name": "PDF",
          "Description": "This is a sample export to PDF",
          "Created": "\\Date(1588776549595)\\",
          "Modified": "\\Date(1588776549595)\\",
          "Visible": true,
          "Deleted": false,
          "IsMoveable": true,
          "Key": "6ad634d0764849b9801600ad8f7fe56b"
        }
      ]
    }
  ],
  ...
}
```

```
    "ResultSuccess": true
  },
  {
    "Ident": "ItemResourceSave",
    "Type": "Insert",
    "ResultVersionKey": "236b612ef1ea4de6842e1cc4fd299e0e",
    "ResultSuccess": true
  }
],
"ResultSuccess": true
}
...
```

Export report template to Destination File Item

Url Structure:

`http://reports.stimulsoft.com/1/reporttemplates/{ReportTemplateKey}/run`

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-DestinationItemKey` contains the unique key of the container element. When creating the container element, it is indicated as the value of the `Key` parameter in response. In PUT-data must specify the JSON-object, which describing the new file item:

PUT-data in the JSON-object

```
...
{
  'FileName': 'ExportToPDF',
  'ExportSet': {
    'Ident': 'Pdf',
    'PageRange': {},
    'EmbeddedFonts': 'false'
    'DitheringType': 'None',
    'PdfACompliance': 'true'
    ...
  }
}
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ea46a3c2c2084439832ea4518d8a5af2" -H "x
```

Returns:

The JSON object contains the field `ResultTaskKey` which holds the key of the export task running in the background. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "ReportRun",
  "AllowNotifications": false,
  "AllowSignals": false,
  "FileType": "ReportSnapshot",
  "ItemVisibility": true,
  "NotificationVisibility": false,
  "ResultTaskKey": "aff29fe55fb54c4d82951dddc7e6d6ef",
  "ResultSuccess": true
}
...
```

1.2.12.2 XPS

There are two stages in the export of a report template:

- > [A file item is created for a future report;](#)
- > [Export a report template to the destination file item.](#)

Create Destination File Item**Url Structure:**

`http://reports.stimulsoft.com/1/files`

Method:

`POST`

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.
A custom header `x-sti-SessionKey` contains the session key of the current user.
In POST-data must specify the JSON-object, which describing the new file item:

POST-data in the JSON-object

```
...
{
```

```
'Ident': 'FormItem',  
'Name': 'XPS',  
'Description': 'This is a sample export to XPS',  
'FileType': 'Xps'  
}  
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: 4ccbebaac38642b4a182d555aa09ee46" -d "
```

Returns:

The JSON object containing the field `ResultSessionKey`, which is a list of members of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...  
{  
  "Ident": "CommandListRun",  
  "ContinueAfterError": false,  
  "ResultCommands": [  
    {  
      "Ident": "ItemSave",  
      "AllowSignalsReturn": false,  
      "SaveEmptyResources": false,  
      "ResultItems": [  
        {  
          "Ident": "FormItem",  
          "FileType": "Xps",  
          "ShareLevel": "Private",  
          "HasItems": false,  
          "StateKey": "1",  
          "WorkspaceKey": "8a87e146d96e4b2e9aa127b22d6d98df",  
          "Name": "XPS",  
          "Description": "This is a sample export to XPS",  
          "Created": "\\Date(1588776549595)\\",  
          "Modified": "\\Date(1588776549595)\\",  
          "Visible": true,  
          "Deleted": false,  
          "IsMoveable": true,  
          "Key": "6ad634d0764849b9801600ad8f7fe56b"  
        }  
      ],  
      "ResultSuccess": true  
    },  
    {  
      "Ident": "ItemResourceSave",  
      "Type": "Insert",  
      "ResultVersionKey": "236b612ef1ea4de6842e1cc4fd299e0e",  
      "ResultSuccess": true  
    }  
  ]  
}
```

```
    },
  ],
  "ResultSuccess": true
}
...
```

Export report template to Destination File Item

Url Structure:

<http://reports.stimulsoft.com/1/reporttemplates/{ReportTemplateKey}/run>

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-DestinationItemKey` contains the unique key of the container element. When creating the container element, it is indicated as the value of the `Key` parameter in response. In PUT-data must specify the JSON-object, which describing the new file item:

PUT-data in the JSON-object

```
...
{
  'FileName': 'XPS',
  'ExportSet': {
    'Ident': 'Xps',
    'PageRange': {},
    'ImageQuality': '75'
    'ImageResolution': '200'
    ...
  }
}
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ea46a3c2c2084439832ea4518d8a5af2" -H "x-sti-DestinationItemKey: 1" -d '{"FileName': 'XPS', 'ExportSet': { 'Ident': 'Xps', 'PageRange': {}, 'ImageQuality': '75', 'ImageResolution': '200' }}
```

Returns:

The JSON object contains the field `ResultTaskKey` which holds the key of the export task running in the background. The success of the command execution is

checked by the content of the field `ResultSuccess`.

Sample **JSON** response

```
...
{
  "Ident": "ReportRun",
  "AllowNotifications": false,
  "AllowSignals": false,
  "FileType": "ReportSnapshot",
  "ItemVisibility": true,
  "NotificationVisibility": false,
  "ResultTaskKey": "aff29fe55fb54c4d82951dddc7e6d6ef",
  "ResultSuccess": true
}
...
```

1.2.12.3 Power Point

There are two stages in the export of a report template:

- [A file item is created for a future report;](#)
- [Export a report template to the destination file item.](#)

Create Destination File Item

Url Structure:

`http://reports.stimulsoft.com/1/files`

Method:

`POST`

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.
A custom header `x-sti-SessionKey` contains the session key of the current user.
In POST-data must specify the JSON-object, which describing the new file item:

POST-data in the **JSON**-object

```
...
{
  'Ident': 'FileItem',
  'Name': 'Power Point',
  'Description': 'This is a sample export to Power Point',
  'FileType': 'PowerPoint'
}
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: 4ccbებაac38642b4a182d555aa09ee46" -d "
```

Returns:

The JSON object containing the field `ResultSessionKey`, which is a list of members of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "CommandListRun",
  "ContinueAfterError": false,
  "ResultCommands": [
    {
      "Ident": "ItemSave",
      "AllowSignalsReturn": false,
      "SaveEmptyResources": false,
      "ResultItems": [
        {
          "Ident": "FileItem",
          "FileType": "PowerPoint",
          "ShareLevel": "Private",
          "HasItems": false,
          "StateKey": "1",
          "WorkspaceKey": "8a87e146d96e4b2e9aa127b22d6d98df",
          "Name": "Power Point",
          "Description": "This is a sample export to Power Point",
          "Created": "\\Date(1588776549595)\\",
          "Modified": "\\Date(1588776549595)\\",
          "Visible": true,
          "Deleted": false,
          "IsMoveable": true,
          "Key": "6ad634d0764849b9801600ad8f7fe56b"
        }
      ],
      "ResultSuccess": true
    },
    {
      "Ident": "ItemResourceSave",
      "Type": "Insert",
      "ResultVersionKey": "236b612ef1ea4de6842e1cc4fd299e0e",
      "ResultSuccess": true
    }
  ],
  "ResultSuccess": true
}
...
```

Export report template to Destination File Item

Url Structure:

`http://reports.stimulsoft.com/1/reporttemplates/{ReportTemplateKey}/run`

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-DestinationItemKey` contains the unique key of the container element. When creating the container element, it is indicated as the value of the `Key` parameter in response. In PUT-data must specify the JSON-object, which describing the new file item:

PUT-data in the JSON-object

```
...
{
  'FileName': 'Power Point',
  'ExportSet': {
    'Ident': 'PowerPoint',
    'PageRange': {},
    'ImageQuality': '75'
    'ImageResolution': '200'
    ...
  }
}
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ea46a3c2c2084439832ea4518d8a5af2" -H "x-  
-d "{ 'FileName': 'Power Point', 'ExportSet': { 'Ident': 'PowerPoint', '
```

Returns:

The JSON object contains the field `ResultTaskKey` which holds the key of the export task running in the background. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
```

```
"Ident": "ReportRun",
"AllowNotifications": false,
"AllowSignals": false,
"FileType": "ReportSnapshot",
"ItemVisibility": true,
"NotificationVisibility": false,
"ResultTaskKey": "aff29fe55fb54c4d82951dddc7e6d6ef",
"ResultSuccess": true
}
...
```

1.2.12.4 HTML

There are two stages in the export of a report template:

- [A file item is created for a future report;](#)
- [Export a report template to the destination file item.](#)

Create Destination File Item

Url Structure:

`http://reports.stimulsoft.com/1/files`

Method:

`POST`

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.
A custom header `x-sti-SessionKey` contains the session key of the current user.
In POST-data must specify the JSON-object, which describing the new file item:

POST-data in the JSON-object

```
...
{
  'Ident': 'FormItem',
  'Name': 'HTML',
  'Description': 'This is a sample export to HTML',
  'FileType': 'Html'
}
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: 4ccbebaac38642b4a182d555aa09ee46" -d "
```

Returns:

The JSON object containing the field `ResultSessionKey`, which is a list of members of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "CommandListRun",
  "ContinueAfterError": false,
  "ResultCommands": [
    {
      "Ident": "ItemSave",
      "AllowSignalsReturn": false,
      "SaveEmptyResources": false,
      "ResultItems": [
        {
          "Ident": "FileItem",
          "FileType": "Html",
          "ShareLevel": "Private",
          "HasItems": false,
          "StateKey": "1",
          "WorkspaceKey": "8a87e146d96e4b2e9aa127b22d6d98df",
          "Name": "HTML",
          "Description": "This is a sample export to HTML",
          "Created": "\\Date(1588776549595)\\",
          "Modified": "\\Date(1588776549595)\\",
          "Visible": true,
          "Deleted": false,
          "IsMoveable": true,
          "Key": "6ad634d0764849b9801600ad8f7fe56b"
        }
      ],
      "ResultSuccess": true
    },
    {
      "Ident": "ItemResourceSave",
      "Type": "Insert",
      "ResultVersionKey": "236b612ef1ea4de6842e1cc4fd299e0e",
      "ResultSuccess": true
    }
  ],
  "ResultSuccess": true
}
...
```

Export report template to Destination File Item**Url Structure:**

<http://reports.stimulsoft.com/1/reporttemplates/{ReportTemplateKey}/run>

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-DestinationItemKey` contains the unique key of the container element. When creating the container element, it is indicated as the value of the `Key` parameter in response. In PUT-data must specify the JSON-object, which describing the new file item:

PUT-data in the JSON-object

```
...
{
  'FileName': 'HTML',
  'ExportSet': {
    'Ident': 'Html',
    'PageRange': {},
    'HtmlType': 'Html5',
    'ImageQuality': '75'
    'ImageResolution': '200'
    ...
  }
}
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ea46a3c2c2084439832ea4518d8a5af2" -H "x-  
-d "{ 'FileName': 'HTML', 'ExportSet': { 'Ident': 'Html', 'PageRange': {
```

Returns:

The JSON object contains the field `ResultTaskKey` which holds the key of the export task running in the background. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "ReportRun",
  "AllowNotifications": false,
  "AllowSignals": false,
  "FileType": "ReportSnapshot",
  "ItemVisibility": true,
  "NotificationVisibility": false,
```

```
"ResultTaskKey": "aff29fe55fb54c4d82951dddc7e6d6ef",  
"ResultSuccess": true  
}  
...
```

1.2.12.5 Text

There are two stages in the export of a report template:

- [A file item is created for a future report;](#)
- [Export a report template to the destination file item.](#)

Create Destination File Item

Url Structure:

`http://reports.stimulsoft.com/1/files`

Method:

POST

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.
A custom header `x-sti-SessionKey` contains the session key of the current user.
In POST-data must specify the JSON-object, which describing the new file item:

POST-data in the JSON-object

```
...  
{  
  'Ident': 'FormItem',  
  'Name': 'Text',  
  'Description': 'This is a sample export to Text',  
  'FileType': 'Text'  
}  
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: 4ccbebaac38642b4a182d555aa09ee46" -d "
```

Returns:

The JSON object containing the field `ResultSessionKey`, which is a list of members of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample **JSON** response

```

...
{
  "Ident": "CommandListRun",
  "ContinueAfterError": false,
  "ResultCommands": [
    {
      "Ident": "ItemSave",
      "AllowSignalsReturn": false,
      "SaveEmptyResources": false,
      "ResultItems": [
        {
          "Ident": "FileItem",
          "FileType": "Text",
          "ShareLevel": "Private",
          "HasItems": false,
          "StateKey": "1",
          "WorkspaceKey": "8a87e146d96e4b2e9aa127b22d6d98df",
          "Name": "Text",
          "Description": "This is a sample export to Text",
          "Created": "\\Date(1588776549595)\\",
          "Modified": "\\Date(1588776549595)\\",
          "Visible": true,
          "Deleted": false,
          "IsMoveable": true,
          "Key": "6ad634d0764849b9801600ad8f7fe56b"
        }
      ],
      "ResultSuccess": true
    },
    {
      "Ident": "ItemResourceSave",
      "Type": "Insert",
      "ResultVersionKey": "236b612ef1ea4de6842e1cc4fd299e0e",
      "ResultSuccess": true
    }
  ],
  "ResultSuccess": true
}
...

```

Export report template to Destination File Item**Url Structure:**

<http://reports.stimulsoft.com/1/reporttemplates/{ReportTemplateKey}/run>

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-DestinationItemKey` contains the unique key of the container element. When creating the container element, it is indicated as the value of the `Key` parameter in response. In PUT-data must specify the JSON-object, which describing the new file item:

PUT-data in the JSON-object

```
...
{
  'FileName': 'Text',
  'ExportSet': {
    'Ident': 'Text',
    'PageRange': {},
    'Encoding': 'UTF8'
    'KillSoaceLines': 'false',
    'PutFeedPageCode': 'false'
    ...
  }
}
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ea46a3c2c2084439832ea4518d8a5af2" -H "x-  
-d "{ 'FileName':'Text', 'ExportSet':{ 'Ident':'Text', 'PageRange': {
```

Returns:

The JSON object contains the field `ResultTaskKey` which holds the key of the export task running in the background. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "ReportRun",
  "AllowNotifications": false,
  "AllowSignals": false,
  "FileType": "ReportSnapshot",
  "ItemVisibility": true,
  "NotificationVisibility": false,
  "ResultTaskKey": "aff29fe55fb54c4d82951dddc7e6d6ef",
  "ResultSuccess": true
}
...
```

1.2.12.6 Rich Text

There are two stages in the export of a report template:

- [A file item is created for a future report;](#)
- [Export a report template to the destination file item.](#)

Create Destination File Item

Url Structure:

```
http://reports.stimulsoft.com/1/files
```

Method:

```
POST
```

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.

A custom header `x-sti-SessionKey` contains the session key of the current user.

In POST-data must specify the JSON-object, which describing the new file item:

POST-data in the JSON-object

```
...
{
  'Ident': 'FormItem',
  'Name': 'Rich Text',
  'Description': 'This is a sample export to RTF',
  'FileType': 'RichText'
}
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: 4ccbebaac38642b4a182d555aa09ee46" -d "
```

Returns:

The JSON object containing the field `ResultSessionKey`, which is a list of members of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
```

```
"Ident": "CommandListRun",
"ContinueAfterError": false,
"ResultCommands": [
  {
    "Ident": "ItemSave",
    "AllowSignalsReturn": false,
    "SaveEmptyResources": false,
    "ResultItems": [
      {
        "Ident": "FileItem",
        "FileType": "RichText",
        "ShareLevel": "Private",
        "HasItems": false,
        "StateKey": "1",
        "WorkspaceKey": "8a87e146d96e4b2e9aa127b22d6d98df",
        "Name": "Rich Text",
        "Description": "This is a sample export to RTF",
        "Created": "\\Date(1588776549595)\\",
        "Modified": "\\Date(1588776549595)\\",
        "Visible": true,
        "Deleted": false,
        "IsMoveable": true,
        "Key": "6ad634d0764849b9801600ad8f7fe56b"
      }
    ],
    "ResultSuccess": true
  },
  {
    "Ident": "ItemResourceSave",
    "Type": "Insert",
    "ResultVersionKey": "236b612ef1ea4de6842e1cc4fd299e0e",
    "ResultSuccess": true
  }
],
"ResultSuccess": true
}
...
```

Export report template to Destination File Item

Url Structure:

<http://reports.stimulsoft.com/1/reporttemplates/{ReportTemplateKey}/run>

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.
A custom header `x-sti-DestinationItemKey` contains the unique key of the container element. When creating the container element, it is indicated as the value

of the `Key` parameter in response. In PUT-data must specify the JSON-object, which describing the new file item:

PUT-data in the JSON-object

```
...
{
  'FileName': 'Rich Text',
  'ExportSet': {
    'Ident': 'RichText',
    'PageRange': {},
    'ExportMode': 'Frame'
    'ImageResolution': 'None',
    'UsePageHeadersAndFooters': 'false'
    ...
  }
}
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ea46a3c2c2084439832ea4518d8a5af2" -H "x-
-d '{"FileName':'Rich Text', 'ExportSet':{ 'Ident':'RichText', 'PageR
```

Returns:

The JSON object contains the field `ResultTaskKey` which holds the key of the export task running in the background. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "ReportRun",
  "AllowNotifications": false,
  "AllowSignals": false,
  "FileType": "ReportSnapshot",
  "ItemVisibility": true,
  "NotificationVisibility": false,
  "ResultTaskKey": "aff29fe55fb54c4d82951dddc7e6d6ef",
  "ResultSuccess": true
}
...
```

1.2.12.7 Word

There are two stages in the export of a report template:

- [A file item is created for a future report;](#)
- [Export a report template to the destination file item.](#)

Create Destination File Item

Url Structure:

`http://reports.stimulsoft.com/1/files`

Method:

`POST`

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.

A custom header `x-sti-SessionKey` contains the session key of the current user.

In POST-data must specify the JSON-object, which describing the new file item:

POST-data in the JSON-object

```
...
{
  'Ident': 'FormItem',
  'Name': 'Word',
  'Description': 'This is a sample export to Word',
  'FileType': 'Word'
}
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: 4ccbebaac38642b4a182d555aa09ee46" -d "
```

Returns:

The JSON object containing the field `ResultSessionKey`, which is a list of members of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "CommandListRun",
  "ContinueAfterError": false,
  "ResultCommands": [
    {
      "Ident": "ItemSave",
      "AllowSignalsReturn": false,
      "SaveEmptyResources": false,
```

```

    "ResultItems": [
      {
        "Ident": "FileItem",
        "FileType": "Word",
        "ShareLevel": "Private",
        "HasItems": false,
        "StateKey": "1",
        "WorkspaceKey": "8a87e146d96e4b2e9aa127b22d6d98df",
        "Name": "Word",
        "Description": "This is a sample export to Word",
        "Created": "\\Date(1588776549595)\\",
        "Modified": "\\Date(1588776549595)\\",
        "Visible": true,
        "Deleted": false,
        "IsMoveable": true,
        "Key": "6ad634d0764849b9801600ad8f7fe56b"
      }
    ],
    "ResultSuccess": true
  },
  {
    "Ident": "ItemResourceSave",
    "Type": "Insert",
    "ResultVersionKey": "236b612ef1ea4de6842e1cc4fd299e0e",
    "ResultSuccess": true
  }
],
"ResultSuccess": true
}
...

```

Export report template to Destination File Item

Url Structure:

<http://reports.stimulsoft.com/1/reporttemplates/{ReportTemplateKey}/run>

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-DestinationItemKey` contains the unique key of the container element. When creating the container element, it is indicated as the value of the `Key` parameter in response. In PUT-data must specify the JSON-object, which describing the new file item:

PUT-data in the JSON-object

```
...
{
  'FileName': 'Word',
  'ExportSet': {
    'Ident': 'Word',
    'PageRange': {},
    'ImageQuality': '75',
    'ImageResolution': '200',
    'RemoveEmptySpaceAtBottom': 'false'
  }
}
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ea46a3c2c2084439832ea4518d8a5af2" -H "x-  
-d '{"FileName':'Word', 'ExportSet':{ 'Ident':'Word', 'PageRange': {}
```

Returns:

The JSON object contains the field `ResultTaskKey` which holds the key of the export task running in the background. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "ReportRun",
  "AllowNotifications": false,
  "AllowSignals": false,
  "FileType": "ReportSnapshot",
  "ItemVisibility": true,
  "NotificationVisibility": false,
  "ResultTaskKey": "aff29fe55fb54c4d82951dddc7e6d6ef",
  "ResultSuccess": true
}
...
```

1.2.12.8 Open Document Writer

There are two stages in the export of a report template:

- > [A file item is created for a future report;](#)
- > [Export a report template to the destination file item.](#)

Create Destination File Item

Url Structure:

`http://reports.stimulsoft.com/1/files`

Method:

POST

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.

A custom header `x-sti-SessionKey` contains the session key of the current user.

In POST-data must specify the JSON-object, which describing the new file item:

POST-data in the JSON-object

```
...
{
  'Ident': 'FormItem',
  'Name': 'ODW',
  'Description': 'This is a sample export to Open Document Writer',
  'FileType': 'OpenDocumentWriter'
}
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: 4ccbebaac38642b4a182d555aa09ee46" -d "
```

Returns:

The JSON object containing the field `ResultSessionKey`, which is a list of members of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "CommandListRun",
  "ContinueAfterError": false,
  "ResultCommands": [
    {
      "Ident": "ItemSave",
      "AllowSignalsReturn": false,
      "SaveEmptyResources": false,
      "ResultItems": [
        {
          "Ident": "FormItem",
          "FileType": "OpenDocumentWriter",
          "ShareLevel": "Private",
          "HasItems": false,

```

```
        "StateKey": "1",
        "WorkspaceKey": "8a87e146d96e4b2e9aa127b22d6d98df",
        "Name": "ODW",
        "Description": "This is a sample export to Open Document
Writer",
        "Created": "\\Date(1588776549595)\\",
        "Modified": "\\Date(1588776549595)\\",
        "Visible": true,
        "Deleted": false,
        "IsMoveable": true,
        "Key": "6ad634d0764849b9801600ad8f7fe56b"
    },
    "ResultSuccess": true
},
{
    "Ident": "ItemResourceSave",
    "Type": "Insert",
    "ResultVersionKey": "236b612ef1ea4de6842e1cc4fd299e0e",
    "ResultSuccess": true
}
],
"ResultSuccess": true
}
...
```

Export report template to Destination File Item

Url Structure:

<http://reports.stimulsoft.com/1/reporttemplates/{ReportTemplateKey}/run>

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-DestinationItemKey` contains the unique key of the container element. When creating the container element, it is indicated as the value of the `Key` parameter in response. In PUT-data must specify the JSON-object, which describing the new file item:

PUT-data in the JSON-object

```
...
{
  'FileName': 'ODW',
  'ExportSet': {
    'Ident': 'OpenDocumentWriter',
```

```
'PageRange': {},  
'ImageQuality': '75',  
'ImageResolution': '200',  
'RemoveEmptySpaceAtBottom': 'false'  
...  
}  
}  
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ea46a3c2c2084439832ea4518d8a5af2" -H "x-  
-d '{"FileName': 'OpenDocumentWriter', 'ExportSet': { 'Ident': 'OpenDocu
```

Returns:

The JSON object contains the field `ResultTaskKey` which holds the key of the export task running in the background. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...  
{  
  "Ident": "ReportRun",  
  "AllowNotifications": false,  
  "AllowSignals": false,  
  "FileType": "ReportSnapshot",  
  "ItemVisibility": true,  
  "NotificationVisibility": false,  
  "ResultTaskKey": "aff29fe55fb54c4d82951dddc7e6d6ef",  
  "ResultSuccess": true  
}  
...
```

1.2.12.9 Excel

There are two stages in the export of a report template:

- > [A file item is created for a future report;](#)
- > [Export a report template to the destination file item.](#)

Create Destination File Item**Url Structure:**

`http://reports.stimulsoft.com/1/files`

Method:

POST

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.
A custom header `x-sti-SessionKey` contains the session key of the current user.
In POST-data must specify the JSON-object, which describing the new file item:

POST-data in the JSON-object

```
...
{
  'Ident': 'FormItem',
  'Name': 'Excel',
  'Description': 'This is a sample export to Excel',
  'FileType': 'Excel'
}
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: 4ccbebaac38642b4a182d555aa09ee46" -d "
```

Returns:

The JSON object containing the field `ResultSessionKey`, which is a list of members of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "CommandListRun",
  "ContinueAfterError": false,
  "ResultCommands": [
    {
      "Ident": "ItemSave",
      "AllowSignalsReturn": false,
      "SaveEmptyResources": false,
      "ResultItems": [
        {
          "Ident": "FormItem",
          "FileType": "Excel",
          "ShareLevel": "Private",
          "HasItems": false,
          "StateKey": "1",
          "WorkspaceKey": "8a87e146d96e4b2e9aa127b22d6d98df",
          "Name": "Excel",
          "Description": "This is a sample export to Excel",
          "Created": "\\Date(1588776549595)\\",

```

```
        "Modified": "\\Date(1588776549595)\\",
        "Visible": true,
        "Deleted": false,
        "IsMoveable": true,
        "Key": "6ad634d0764849b9801600ad8f7fe56b"
      }
    ],
    "ResultSuccess": true
  },
  {
    "Ident": "ItemResourceSave",
    "Type": "Insert",
    "ResultVersionKey": "236b612ef1ea4de6842e1cc4fd299e0e",
    "ResultSuccess": true
  }
],
"ResultSuccess": true
}
```

Export report template to Destination File Item

Url Structure:

<http://reports.stimulsoft.com/1/reporttemplates/{ReportTemplateKey}/run>

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-DestinationItemKey` contains the unique key of the container element. When creating the container element, it is indicated as the value of the `Key` parameter in response. In PUT-data must specify the JSON-object, which describing the new file item:

PUT-data in the JSON-object

```
...
{
  'FileName': 'ExportReport.xlsx',
  'ExportSet': {
    'Ident': 'Excel2007',
    'ImageQuality': '75'
    ...
  }
}
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: 7cc93e33645f4e05975e3a468229c00f" -H "x-  
-d "{ 'FileName': 'ExportReport.xlsx', 'ExportSet': { 'Ident': 'Excel200
```

Returns:

The JSON object contains the field `ResultTaskKey` which holds the key of the export task running in the background. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...  
{  
  "Ident": "ReportRun",  
  "AllowNotifications": false,  
  "AllowSignals": false,  
  "FileType": "ReportSnapshot",  
  "ItemVisibility": true,  
  "NotificationVisibility": false,  
  "ResultTaskKey": "aff29fe55fb54c4d82951dddc7e6d6ef",  
  "ResultSuccess": true  
}  
...
```

1.2.12.10 Open Document Calc

There are two stages in the export of a report template:

- [A file item is created for a future report;](#)
- [Export a report template to the destination file item.](#)

Create Destination File Item**Url Structure:**

`http://reports.stimulsoft.com/1/files`

Method:

POST

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.
A custom header `x-sti-SessionKey` contains the session key of the current user.
In POST-data must specify the JSON-object, which describing the new file item:

POST-data in the JSON-object

```
...
{
  'Ident': 'FormItem',
  'Name': 'ODC',
  'Description': 'This is a sample export to Open Document Calc',
  'FileType': 'OpenDocumentCalc'
}
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: 4ccbebaac38642b4a182d555aa09ee46" -d "
```

Returns:

The JSON object containing the field `ResultSessionKey`, which is a list of members of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "CommandListRun",
  "ContinueAfterError": false,
  "ResultCommands": [
    {
      "Ident": "ItemSave",
      "AllowSignalsReturn": false,
      "SaveEmptyResources": false,
      "ResultItems": [
        {
          "Ident": "FormItem",
          "FileType": "OpenDocumentCalc",
          "ShareLevel": "Private",
          "HasItems": false,
          "StateKey": "1",
          "WorkspaceKey": "8a87e146d96e4b2e9aa127b22d6d98df",
          "Name": "ODC",
          "Description": "This is a sample export to Open Document Calc",
          "Created": "\\Date(1588776549595)\\",
          "Modified": "\\Date(1588776549595)\\",
          "Visible": true,
          "Deleted": false,
          "IsMoveable": true,
          "Key": "6ad634d0764849b9801600ad8f7fe56b"
        }
      ]
    },
    "ResultSuccess": true
  ],
}
```

```
    "Ident": "ItemResourceSave",
    "Type": "Insert",
    "ResultVersionKey": "236b612ef1ea4de6842e1cc4fd299e0e",
    "ResultSuccess": true
  }
],
"ResultSuccess": true
}
...
```

Export report template to Destination File Item

Url Structure:

<http://reports.stimulsoft.com/1/reporttemplates/{ReportTemplateKey}/run>

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.

A custom header `x-sti-DestinationItemKey` contains the unique key of the container element. When creating the container element, it is indicated as the value of the `Key` parameter in response. In PUT-data must specify the JSON-object, which describing the new file item:

PUT-data in the JSON-object

```
...
{
  'FileName': 'ODC',
  'ExportSet': {
    'Ident': 'OpenDocumentCalc',
    'PageRange': {},
    'ImageQuality': '100'
    'ImageResolution': '200'
    ...
  }
}
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ea46a3c2c2084439832ea4518d8a5af2" -H "x-sti-DestinationItemKey: ..." -d "{ 'FileName': 'Open Document Calc', 'ExportSet': { 'Ident': 'OpenDoc
```

Returns:

The JSON object contains the field `ResultTaskKey` which holds the key of the export task running in the background. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "ReportRun",
  "AllowNotifications": false,
  "AllowSignals": false,
  "FileType": "ReportSnapshot",
  "ItemVisibility": true,
  "NotificationVisibility": false,
  "ResultTaskKey": "aff29fe55fb54c4d82951dddc7e6d6ef",
  "ResultSuccess": true
}
...
```

1.2.12.11 Data

There are two stages in the export of a report template:

- [A file item is created for a future report;](#)
- [Export a report template to the destination file item.](#)

Create Destination File Item**Url Structure:**

`http://reports.stimulsoft.com/1/files`

Method:

`POST`

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.
A custom header `x-sti-SessionKey` contains the session key of the current user.
In POST-data must specify the JSON-object, which describing the new file item:

POST-data in the JSON-object

```
...
{
  'Ident': 'FileItem',
  'Name': 'Data',

```

```
'Description': 'This is a sample export to Data (CSV)',  
'FileType': 'Data'  
}  
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: 4ccbebaac38642b4a182d555aa09ee46" -d "
```

Returns:

The JSON object containing the field `ResultSessionKey`, which is a list of members of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...  
{  
  "Ident": "CommandListRun",  
  "ContinueAfterError": false,  
  "ResultCommands": [  
    {  
      "Ident": "ItemSave",  
      "AllowSignalsReturn": false,  
      "SaveEmptyResources": false,  
      "ResultItems": [  
        {  
          "Ident": "FileItem",  
          "FileType": "Data",  
          "ShareLevel": "Private",  
          "HasItems": false,  
          "StateKey": "1",  
          "WorkspaceKey": "8a87e146d96e4b2e9aa127b22d6d98df",  
          "Name": "Data",  
          "Description": "This is a sample export to Data (CSV)",  
          "Created": "\\Date(1588776549595)\\",  
          "Modified": "\\Date(1588776549595)\\",  
          "Visible": true,  
          "Deleted": false,  
          "IsMoveable": true,  
          "Key": "6ad634d0764849b9801600ad8f7fe56b"  
        }  
      ],  
      "ResultSuccess": true  
    },  
    {  
      "Ident": "ItemResourceSave",  
      "Type": "Insert",  
      "ResultVersionKey": "236b612ef1ea4de6842e1cc4fd299e0e",  
      "ResultSuccess": true  
    }  
  ],  
}
```

```
"ResultSuccess": true
}
...
```

Export report template to Destination File Item

Url Structure:

`http://reports.stimulsoft.com/1/reporttemplates/{ReportTemplateKey}/run`

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-DestinationItemKey` contains the unique key of the container element. When creating the container element, it is indicated as the value of the `Key` parameter in response. In PUT-data must specify the JSON-object, which describing the new file item:

PUT-data in the JSON-object

```
...
{
  'FileName': 'FormItem',
  'ExportSet': {
    'Ident': 'Pdf',
    'PageRange': {},
    'DataType': 'Csv'
    'DataExportMode': 'DataAndHeadersFooters'
    ...
  }
}
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ea46a3c2c2084439832ea4518d8a5af2" -H "x-sti-DestinationItemKey: Data" -d '{"FileName": "Data", "ExportSet": {"Ident": "Data", "PageRange": {}}}
```

Returns:

The JSON object contains the field `ResultTaskKey` which holds the key of the export task running in the background. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample **JSON** response

```
...
{
  "Ident": "ReportRun",
  "AllowNotifications": false,
  "AllowSignals": false,
  "FileType": "ReportSnapshot",
  "ItemVisibility": true,
  "NotificationVisibility": false,
  "ResultTaskKey": "aff29fe55fb54c4d82951dddc7e6d6ef",
  "ResultSuccess": true
}
...
```

1.2.12.12Image

There are two stages in the export of a report template:

- > [A file item is created for a future report;](#)
- > [Export a report template to the destination file item.](#)

Create Destination File Item

Url Structure:

`http://reports.stimulsoft.com/1/files`

Method:

`POST`

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.

A custom header `x-sti-SessionKey` contains the session key of the current user.

In POST-data must specify the JSON-object, which describing the new file item:

POST-data in the **JSON**-object

```
...
{
  'Ident': 'FormItem',
  'Name': 'Image',
  'Description': 'This is a sample export to image',
  'FileType': 'Image'
}
...
```

CURL example:

```
curl -X POST -H "x-sti-SessionKey: 4ccbebaac38642b4a182d555aa09ee46" -d "
```

Returns:

The JSON object containing the field `ResultSessionKey`, which is a list of members of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "CommandListRun",
  "ContinueAfterError": false,
  "ResultCommands": [
    {
      "Ident": "ItemSave",
      "AllowSignalsReturn": false,
      "SaveEmptyResources": false,
      "ResultItems": [
        {
          "Ident": "FileItem",
          "FileType": "Image",
          "ShareLevel": "Private",
          "HasItems": false,
          "StateKey": "1",
          "WorkspaceKey": "8a87e146d96e4b2e9aa127b22d6d98df",
          "Name": "Image",
          "Description": "This is a sample export to image",
          "Created": "\\Date(1588776549595)\\",
          "Modified": "\\Date(1588776549595)\\",
          "Visible": true,
          "Deleted": false,
          "IsMoveable": true,
          "Key": "6ad634d0764849b9801600ad8f7fe56b"
        }
      ],
      "ResultSuccess": true
    },
    {
      "Ident": "ItemResourceSave",
      "Type": "Insert",
      "ResultVersionKey": "236b612ef1ea4de6842e1cc4fd299e0e",
      "ResultSuccess": true
    }
  ],
  "ResultSuccess": true
}
...
```

Export report template to Destination File Item

Url Structure:

`http://reports.stimulsoft.com/1/reporttemplates/{ReportTemplateKey}/run`

Method:

PUT

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. A custom header `x-sti-DestinationItemKey` contains the unique key of the container element. When creating the container element, it is indicated as the value of the `Key` parameter in response. In PUT-data must specify the JSON-object, which describing the new file item:

PUT-data in the JSON-object

```
...
{
  'FileName': 'Image',
  'ExportSet': {
    'Ident': 'Image',
    'ImageType': 'Png',
    'ImageResolution': '200'
    ...
  }
}
...
```

CURL example:

```
curl -X PUT -H "x-sti-SessionKey: ea46a3c2c2084439832ea4518d8a5af2" -H "x-sti-DestinationItemKey: " -d "{ 'FileName': 'Image', 'ExportSet': { 'Ident': 'Image', 'ImageType': 'Png', 'ImageResolution': '200' } }
```

Returns:

The JSON object contains the field `ResultTaskKey` which holds the key of the export task running in the background. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

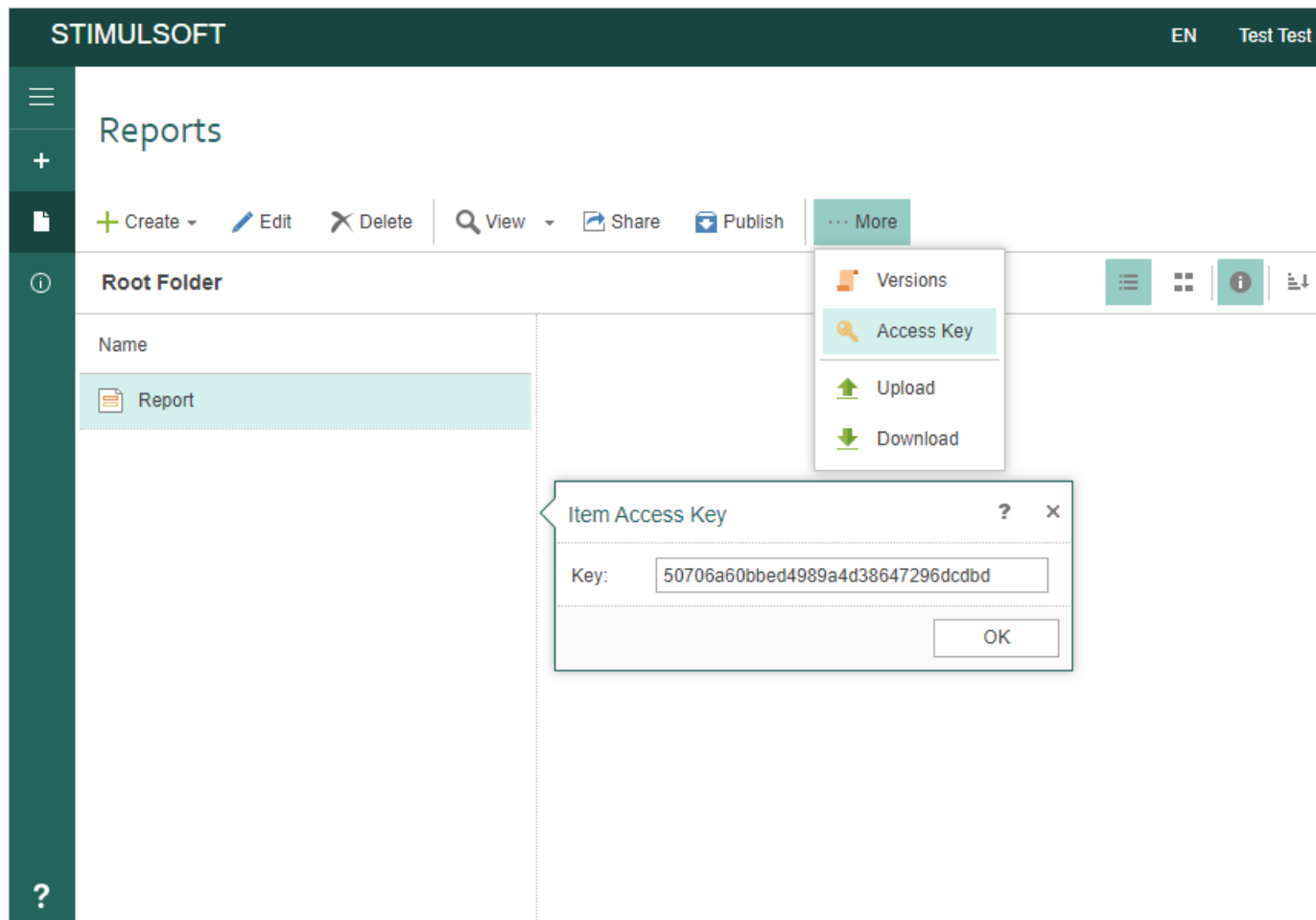
```
...
{
  "Ident": "ReportRun",
  "AllowNotifications": false,
  "ResultTaskKey": "ReportRun",
  "ResultSuccess": true
}
```

```
"AllowSignals": false,
"FileType": "ReportSnapshot",
"ItemVisibility": true,
"NotificationVisibility": false,
"ResultTaskKey": "aff29fe55fb54c4d82951dddc7e6d6ef",
"ResultSuccess": true
}
...
```

1.2.12.13 Download

Description:

You can download any element. For this, you need to know its unique key. You can [get the Item key using the Access Key command](#) in your [Stimulsoft Cloud account](#).



Url Structure:

<http://reports.stimulsoft.com/1/files/{ItemKey}>

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: 22ed00099bd24fffacf9d5ad2344f457" http:
```

Returns:

The JSON object containing the collection `ResultItems`, which contains a list of items in the specified folder of the current workspace. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "CommandListRun",
  "ContinueAfterError": false,
  "ResultCommands": [
    {
      "Ident": "ItemGet",
      "AllowDeleted": false,
      "ResultItem": {
        "Ident": "FileItem",
        "FileType": "Pdf",
        "ShareLevel": "Private",
        "HasItems": false,
        "StateKey": "3",
        "WorkspaceKey": "8a87e146d96e4b2e9aa127b22d6d98df",
        "Name": "Test",
        "Description": "This is a TEST",
        "Created": "\\Date(1588776549597)\\",
        "Modified": "\\Date(1588776602287)\\",
        "Visible": true,
        "Deleted": false,
        "IsMoveable": true,
        "Key": "6ad634d0764849b9801600ad8f7fe56b"
      },
      "ResultLastVersionKey": "87ea176df38346209cb4561ad86a8840",
      "ResultSuccess": true
    },
    {
      "Ident": "ItemResourceGet",
      "ResultResource": "JVBERi0xLjcNCiXi48/
TDQoxIDAgb2JqDD.....", //your pdf file content
      "ResultSuccess": true
    }
  ],
}
```

```
"ResultSuccess": true
}
...
```

1.2.12.14 Track Task Status

Description:

Getting the status of a task that is running in the background.

Url Structure:

`http://reports.stimulsoft.com/1/task/{taskKey}`

Method:

GET

Parameters:

A custom header `x-sti-SessionKey` contains the session key of the current user. The URL contains the task key for tracking its status.

CURL example:

```
curl -X GET -H "x-sti-SessionKey: 1add6a4f1c5e481c80e964b613ee6089" http:
```

Returns:

The JSON object containing the collection `ResultStatus`, which contains a list of the task information. The success of the command execution is checked by the content of the field `ResultSuccess`.

Sample JSON response

```
...
{
  "Ident": "TaskStatus",
  "ResultStatus": {
    "Ident": "Task",
    "Name": "ReportRunAndTransfer",
    "Created": "2023-09-08T09:24:56.283Z",
    "Started": "2023-09-08T09:24:56.34Z",
    "Status": "Running",
    "IsMonitorTask": false,
    "IsWaiting": false,
    "IsRunning": true,
    "IsProcessed": false,
    "IsFinished": false,
    "IsStopped": false,
    "IsError": false
  },
  "ResultSuccess": true
}
```

...