Creatio Academy

Portal

Self-service portal

Version 8.0

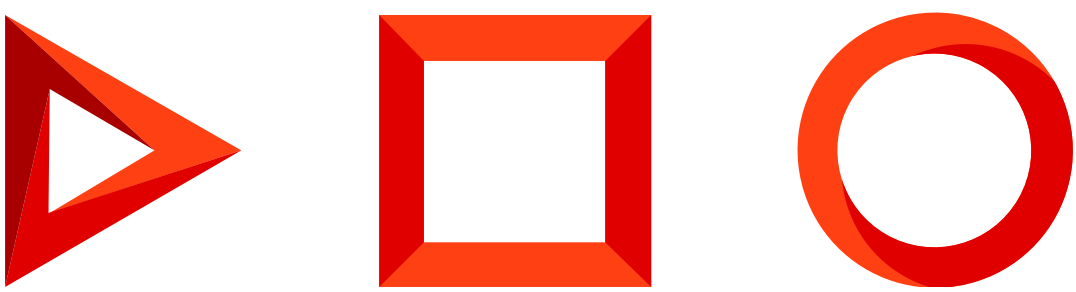

Table of Contents

Self-service portal

Advanced

The Self-service Portal (SSP) is an integral part of the Service Creatio, enterprise edition and Creatio customer center products, as well as all bundles that these products are part of.

The SSP is a workplace where portal users are automatically redirected after login.

Portal users have access to the [*Cases*] and [*Knowledge base*] sections and to the [*Self-service portal main page*], which contains general summary information and works as a single workplace for a portal user.

The [*Cases*] section is used for registration of cases by the customers, viewing the status of their cases, entering additional case information and for obtaining information about case resolution process. By default, a portal user has access to those cases where this user is specified as a contact. The user can enter additional information, publish messages and interact with the service staff on the case page. The case history is displayed on the case page at the [*Processing*] tab.

The portal's [*Knowledge base*] section is used for searching for reference information or a solution. This section can be filled only by the helpdesk staff from the main interface of the system.

Portal interface

From the development point of view, the portal is a preconfigured separate workplace. By default, this workplace is not available for the ordinary portal users. A system user with the "portal user" type automatically enters this workplace (the portal main page) after authorization.

Portal sections and pages are the same as sections and system edit pages from the main system interface and they work with the same `Entities`. The case edit page on the portal is simpler compared to the regular case page and does not contain the majority of fields. These edit pages are different objects in configuration (`CasePage` and `PortalCasePage`).

The system of portal access permissions slightly differs. To grant access to the specific entities (EntitySchema) for the portal users, you need to specify these entities in the [*List of objects available for portal users*] lookup. Self-service portal licenses limit the number of records that can be added to this lookup. By default, the number is limited to 70 records.

Working with the page wizard on the portal

The portal user cannot access the functions of the page-, detail- and section wizards These functions can be accessed from the main system interface with administrator permissions in the following way:

1. Enter the [*Workplace setup*] section in the system designer.
2. Select the [*Portal*] workplace and click the [*Open*] button.
3. Select the required section and click the [*Section wizard*] button.

The standard section wizard will open.

Configuring the portal and portal users

To start using the portal:

1. Ensure that the `/configuration/terrasoft/auth` option in the `web.config` file of the application loader (the "external" `web.config`) has the following in it:

   `/configuration/terrasoft/auth` option

   ```
   <terrasoft>
       auth providerNames="InternalUserPassword,SSPUserPassword" …>
       ...
   </terrasoft>
   ```

   This setting is responsible for the authorization in the portal users in the system.
2. Create a contact for the user.
3. Create a user with the "Portal user" type. Fill out the required fields.
4. Provide all necessary licenses for the user.

A portal user is required to have a valid time zone specified in the profile. The time zone is not specified for new users by default. Portal users must edit their profiles and select the proper time zone The system will display all dates and times in the portal user's local time.

Restricting access to web services for portal users

The portal is a Creatio platform component whose main purpose is to implement self-service processes for your customers and partners. You can also use the portal to organize the work of internal users if you need to restrict access permissions to functionality of the primary application.

Using portal enables many external users (who are not employees of your organization) to access some of the Creatio data. For this, you need to manage which users (portal or company employees) have access to the application web services.

Attention. This article is valid for applicaation version 7.13.2 and up.

Route prefixes for web services configuration

Route prefixes in Creatio enable managing access to the application web services. You can set the needed route prefix using specific `ServiceRoute` attributes of the service class.

Route prefixes for configuration web services

Access level	Attribute	Route prefix	Example of code
For self-	`SspServiceRoute`	`/ssp`	`[ServiceContract]`

Access level	Attribute	Route prefix	Example of code

service portal users only			[ServiceContract]
			[DefaultServiceRoute]
			public class SspOnlyService : BaseServic
			{
			}
			
			Example of call
			`~/ssp/rest/SspOnlyService`
			`~/ssp/soap/SspOnlyService`

For internal users only	`DefaultServiceRoute`
	or
	no `ServiceRoute` attribute is specified
		none	[ServiceContract]
			public class InternalService : BaseServi
			{
			}
			
			or
			
			[ServiceContract]
			[DefaultServiceRoute]
			public class InternalService : BaseServi
			{
			}
			
			Example of call
			`~/rest/InternalService`
			`~/soap/InternalService`

For both: the internal and portal users	Both the
	`DefaultServiceRoute`
	and
	`SspServiceRoute`
		`/ssp`
		or
		none
			[ServiceContract]
			[DefaultServiceRoute]
			[SspServiceRoute]
			public class CommonService : BaseService
			{
			}
			
			Example of call
			`~/rest/CommonService`
			`~/soap/CommonService`
			`~/ssp/rest/CommonService`

Access level	Attribute	Route prefix	Example of code

	The `ServiceRoute` attribute with the specified prefix (e.g., "`custom`") [*ServiceRoute("custom")*]	Arbitrary route prefix of the service	`[ServiceContract]`
			`[ServiceRoute("custom")]`
			`public class CustomPrefixService : BaseS`
			`{`
			`}`
			
			Example of call
			`~/custom/rest/CustomPrefixService`
			`~/custom/soap/CustomPrefixService`

Note. You can use several attributes at the same time: `ServiceRoute`, `SspServiceRoute` and `DefaultServiceRoute`. As a result, the routes for services with all prefix variants will be created.

Restricting access to the internal API for portal users

If a portal user (`SspUserConnection`) addresses a service with a route that does not contain the "`/ssp`" prefix, the service will return error 403 (`Forbidden`).

Restricting access to the portal API for internal users

If an internal user (`UserConnection`) addresses the service with the route that contains the "`/ssp`" prefix, the service will return a page with code 403 (`Forbidden`).

The ServiceStack core services

The `ServiceStack` core services (`DataService`, `ManagerService`, etc.) are available by default to the internal users only.

To make any of these methods available on the portal, tag such method with the `SspServiceAccess` attribute – in this case, the method will have an additional route with the `~/DataService/ssp/…` prefix.

If you need to have a different logic for the portal, create a new service by specifying the `SspServiceAccess` attribute for it and pass the name of the original method to its constructor. Example:

SspServiceAccess attribute

```
[SspServiceAccess(nameof(SelectQuery))]
public class SspSelectQuery : SelectQuery
{
```

```
}
```

This service creates a contract whose method will be registered at the following path: `~/DataService/ssp/SelectQuery`

Access to the `ServiceStack` methods with the “`ssp`” prefix is denied to the internal users. Access to the `ServiceStack` methods without the “`ssp`” prefix is denied to the portal users.

Example of using the PortalMessagePublisherExtensions mixin

 Advanced

Example. Using the PortalMessagePublisherExtensions mixin.

Example implementation

CaseSectionActionsDashboard

```
define("CaseSectionActionsDashboard", ["PortalMessagePublisherExtensions"], function() {
    return {
        mixins: {
            /**
             * @class PortalMessagePublisherExtensions extends tabs and publishers configs.
             */
            PortalMessagePublisherExtensions: "Terrasoft.PortalMessagePublisherExtensions"
        },
        methods: {
            /**
             * @inheritdoc Terrasoft.SectionActionsDashboard#getExtendedConfig
             * @overridden
             */
            getExtendedConfig: function() {
                // Getting the tab configuration object from the parent method.
                var config = this.callParent(arguments);
                // Calling the mixin method, adding a portal tab configuration.
                this.mixins.PortalMessagePublisherExtensions.extendTabsConfig.call(this, config)
                // Returns the extended configuration object.
                return config;
            },

            /**
             * @inheritdoc Terrasoft.SectionActionsDashboard#getSectionPublishers
```

```
             * @overridden
             */
            getSectionPublishers: function() {
                // Getting a collection of message publishers from the parent method.
                var publishers = this.callParent(arguments);
                // Calling the mixin method, adding a portal channel.
                this.mixins.PortalMessagePublisherExtensions.extendSectionPublishers.call(this,
                // Returns the extended collection of message publishers.
                return publishers;
            }
    },
        diff: /**SCHEMA_DIFF*/[
            {
                "operation": "insert",
                "name": "PortalMessageTab",
                "parentName": "Tabs",
                "propertyName": "tabs",
                "values": {
                    "items": []
                }
            },
            {
                "operation": "insert",
                "name": "PortalMessageTabContainer",
                "parentName": "PortalMessageTab",
                "propertyName": "items",
                "values": {
                    "itemType": this.Terrasoft.ViewItemType.CONTAINER,
                    "classes": {
                        "wrapClassName": ["portal-message-content"]
                    },
                    "items": []
                }
            },
            {
                "operation": "insert",
                "name": "PortalMessageModule",
                "parentName": "PortalMessageTab",
                "propertyName": "items",
                "values": {
                    "classes": {
                        "wrapClassName": ["portal-message-module", "message-module"]
                    },
                    "itemType": this.Terrasoft.ViewItemType.MODULE,
                    "moduleName": "PortalMessagePublisherModule",
                    "afterrender": {
                        "bindTo": "onMessageModuleRendered"
                    },
                    "afterrerender": {
```

```
                        "bindTo": "onMessageModuleRendered"
                    }
                }
            }
        ]/**SCHEMA_DIFF*/
    };
});
```

Change access to web service for portal users

Advanced

The application has a set of base services and only internal users have access to them.

To change access to the base service:

1. In the custom package, create a service with the access settings for portal users.
2. In the service, add the base service methods that must be available for portal users.
3. Change the custom or extend the base client schemas by changing the call of base service to the call of the created service (see step 1).
4. Compile the configuration.

Example. Change access to web service for portal users.

Example implementation

Example of the service source code that extends access to the `ActivityUtilService` base service:

PartnerActivityUtilService

```
namespace Terrasoft.Configuration
{
    using System;
    using System.IO;
    using System.Runtime.Serialization;
    using System.ServiceModel;
    using System.ServiceModel.Activation;
    using System.ServiceModel.Web;
    using Terrasoft.Configuration.FileUpload;
    using Terrasoft.Core.Factories;
    using Terrasoft.Web.Common;
    using Terrasoft.Web.Common.ServiceRouting;
```

```
    [ServiceContract]
    // The service is available for both: the internal and portal users
    [DefaultServiceRoute]
    [SspServiceRoute]
    [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Requ
    public class PartnerActivityUtilService: BaseService {
        // Base service, access needs to be extended
        private static readonly ActivityUtilService _baseService = new ActivityUtilService();

        [OperationContract]
        [WebInvoke(Method = "POST", BodyStyle = WebMessageBodyStyle.Wrapped, RequestFormat = Web
        public Guid CreateActivityFileEntity(JsonActivityFile jsonActivityFile) {
            return _baseService.CreateActivityFileEntity(jsonActivityFile);
        }
        [OperationContract]
        [WebInvoke(Method = "POST", BodyStyle = WebMessageBodyStyle.Wrapped, RequestFormat = Web
        public Guid CreateFileEntity(JsonEntityFile jsonEntityFile) {
            return _baseService.CreateFileEntity(jsonEntityFile);
        }
    }
}
```

PortalMessagePublisherExtensions mixin JS

Advanced

A mixin is a class designed to extend the functions of other classes. Mixins are separately created classes with additional functionality. Learn more about mixins in the "Mixins. The "mixins" property" article.

The `PortalMessagePublisherExtensions` mixin is used for the extension of the `SectionActionsDashboard` schema (and its derived schemas). It allows you to extend the configuration of the `SectionActionsDashboard` tabs with the portal message tab `PortalMessageTab` and add the corresponding Portal message `portal`. The mixin is implemented in the [*PortalMessagePublisher*] package and is available in the Service Enterprise product (or in the bundles that include this product).

Methods

`extendTabsConfig(config) : Object`

Extends the configuration of the `SectionActionsDashboard` tabs with the portal messages tab `PortalMessageTab`.

Returns the augmented object (`Object`) of the `SectionActionsDashboard` tab configuration.

The `config` parameter (`Object`) – `SectionActionsDashboard` tab configuration object.

`extendSectionPublishers(publishers) : Array`

Adds a portal channel (`Portal`) to the message publisher collection.

Returns the augmented collection of message publishers (`Array`).

The `publishers` parameter `(Array)` is the collection of message publishers.