CMDB

API-ASS Integration

This document is presented as a general guide to integration with the Aranda Self Service (ASS) console. Some operations are presented in detail including request data, parameters, responses, and error codes.

General

General

Below are the generalities that must be taken into account for the proper functioning of SSA services.

- Supported Database Version 8.0.66
- Operations on the API must be performed with users who have the corresponding permissions assigned from the Aranda PROFILE console.
- The display of the Articles will depend on the previous administration of an Administrator. Only Items that are in an "Approved" status and have an associated "Category" will be displayed.
- The associated Categories must have the Items option selected.
- The services created come with the Aranda Self Service (ASS) installer and are for viewing and querying only.

Description of operations

Description of operations

This section then describes in detail how each of the services corresponding to each method is consumed. Details such as URIs, operation type, parameters (required and optional) and their types, responses, and error codes and messages are included.

Session Management

Session Management

This section describes the operations related to session handling (user authentication and logoff).

Login

Details of the request:

- URI: api/v8.1/user/login
- Type: POST
- Required headings:
- Content-Type: application/json

Parameters:

Name	Guy	Obligatory	Description
USERNAME	Text	Yes	User who will log in.
PASSWORD	Text	Yes	Password corresponding to the user.
LANGUAGEID	Number	No	Language in which the session will be recorded. Possible options are: 1. English 2. Spanish 3. Portuguese If it is not provided, it is recorded in English by default.

Body of the petition:

The request consists of a field-value json array with the possible parameters as follows: Required. Example:

```
[
  {
    "Field": "username",
    "Value": "USUARIO_ASDK"
  },
  {
    "Field": "password",
    "Value": "CONTRASEÑA_DEL_USUARIO"
  }
]
```

Answer:

As an answer, a json object is obtained, with the following information:

```
[
  {
    "Field": "userId",
    "Value": "####"
  },
  {
    "Field": "sessionId",
    "Value": ""###AAABBBCCC###"
  },
  {
    "Field": "result",
    "Value": "True"
  }
]
```


Error messages:

Code	HTTP status	Error Message
400	BadRequest	InvalidUserName
400	BadRequest	InvalidPassword
400	BadRequest	InvalidLangId
401	Unauthorized	InvalidUserData
500	InternalServerError	FailureOnLogin

Logging Out

Details of the request:

- URI: api/v8.1/user/logout
- Type: POST
- Required headings:
- Content-Type: application/json
- Authorization: TOKEN

Parameters:

It is only required to make a request by sending the authentication token as a header.

Body of the petition:

Empty.

Answer:

There is no response body, the validation of the result of the operation can be performed by analyzing the http code obtained: 200 OK for a successful operation.

Error messages:

Code	HTTP status	Error Message
401	Unauthorized	InvalidToken
500	InternalServerError	FailureCloseSession

Item Handling

Item Handling

This section describes the operations related to the handling of the items (Listing, Details, Attachments, and Related).

List of articles

Details of the request:

- URI: api/v8.1/article/list
- Type: POST
- Required headings:
- Content-Type: application/json
- Authorization: TOKEN

Parameters:

Name	Guy	Obligatory	Description
CategoryId	Numerical	No	Category id.
FieldSearch	String	No	Type of search.
PrivacyId	Numerical	Yes	Article privacy.
ProjectId	Numerical	Yes	Project ID.
TextSearch	String	No	Text to search.
TypeId	Numerical	No	Id Item type.

Body of the petition:

The body of the request must contain the PrivacyId and ProjectId parameters, at least to make the query and in turn be filled in.

The ProjectId field only receives integer data and must be filled in with the ID of the project that will be queried.

The PrivacyId field handles privacy and must be filled in with the following integer values:

- 0, See private and public articles.
- 1. Check public articles.

Example

```
{
   "PrivacyId":0,
   "ProjectId":1
}
```

For more specific queries, the following parameters can be added:

- CategoryId, the ID of the category to be consulted must be entered.
- TextSearch, see specific text in the Article, which should be enclosed in quotation marks “”.
- TypeId, the item “Type” id must be entered.
- The parameter FieldSearch, you can send empty or with the following mappings to perform queries according to their type:
- LastVisited: Last visited.
- FavoriteByUser: User Favorites.
- QualifiedByUser: User-rated.
- MostVisitedByUser: Most visited by the user.
- AddedByProject: Added by project.
- MostVisitedByProject: Most visited per project.
- QualifiedByProject: Graded by project.
- AddedByCategory: Added by category.
- MostVisitedByCategory: Most visited by category.
- QualifiedByCategory: Rated by category.

Example:

```
{
   "CategoryId":0,
   "FieldSearch":"QualifiedByUser",
   "PrivacyId":1,
   "ProjectId":2,
   "TextSearch":"Ingresar texto",
   "TypeId":3
}
```

Answer:

The service will return a Json with the following structure:

```
[
 {
   "Description": "Descripción del artículo",
   "Id": valor numérico correspondiente al id,
   "Title": "Titulo del Articulo"
 }
 ]
```

Error messages:

Code	HTTP status	Error Message
400	BadRequest	DataSearchIsNull
400	BadRequest	PrivacyIdIsNull
400	BadRequest	InvalidProjectId
400	BadRequest	InvalidPrivacyId
401	Unauthorized	InvalidToken
500	InternalServerError	FailureSearchArticle

Item Detail

The consultation of the detail of the Item will be made depending on the ID that is sent in the consultation.

Details of the request:

- URI: api/v8.1/article/{id}/detail
- Type: GET
- Required headings:
- Content-Type: application/json
- Authorization: TOKEN

Parameters:

Name	Data type	Obligatory	Description
Id	Number	Yes	Article ID

Body of the petition:

Empty.

Answer:

The service will return a Json with the following structure:


```
{
  "Description": "Descripción del artículo",
  "Id": 1,"Title": "Titulo del Artículo",
  "AuthorId": null,"AuthorName": null,
  "AuthorUserName": null,"CategoryId": null,
  "CategoryName": null,"ClassId": 0,
  "Content": Contenido Html,
  "CreateReasonId": 0,
  "Created": "/Date(1336768613863-0500)/",
  "GroupId": null,
  "InterfaceId": null,
  "IsRatedByUser": true,
  "IsUserFavorite": false,
  "Ispublic": false,
  "KeyWords": "Palabras Claves del artículo",
  "LastModified": "/Date(1336768922067-0500)/",
  "ModifiedDate": null,
  "ModifierUserId": null,
  "ModifierUserName": null,
  "OpenedDate": null,
  "ProjectId": 7,
  "ProjectName": "PPQA",
  "Rated": 1,
  "RatingTotal": 5.5,
  "ReasonDescription": null,
  "ReasonId": 0,
  "ResponsibleGroupId": 0,
  "ResponsibleGroupName": null,
  "ResponsibleId": 0,
  "ResponsibleName": null,
  "SolutionId": null,
  "StatusDescription": null,
  "StatusId": 0,
  "TotalHit": 12,
  "TypeDescription": null,
  "TypeId": 5,
  "TypeName": null,
  "Url": ""
}
```


Error messages:

Code	HTTP status	Error Message
400	BadRequest	InvalidArticleId
401	Unauthorized	InvalidToken
404	NotFound	InvalidArticleId
500	InternalServerError	FailureGetDetailArticle

Attachments to the article

The query of the Attachments associated with an Article is carried out depending on the ID that is sent in the query.

Details of the request:

- URI: api/v8.1/article/{id}/file/list
- Type: GET
- Required headings:
- Content-Type: application/json
- Authorization: TOKEN

Parameters:

Name	Data type	Obligatory	Description
Id	Number	Yes	Article ID

Body of the petition:

Empty.

Answer:

Depending on the number of attachments associated with the Article, the service will return a block for each attachment in a Json with the following structure:

```
[
 {
 "Created": "/Date(1473801979043-0500)/",
 "Id": 3721,
 "Name": "Nombre del adjunto.docx",
 "Size": 634311,
 "Url": "Ubicación del adjunto"
 }
 ]
```

Error messages:

Code	HTTP status	Error Message
400	BadRequest	InvalidArticleId
401	Unauthorized	InvalidToken
404	NotFound	InvalidArticleId
500	InternalServerError	FailureListFiles

Related Posts

The query of the related Items will be made on the ID of the Item that is sent. The idPrivacity field handles the privacy of related articles and must be filled with the following integer values:

- 0, See articles related to private status.
- 1, See articles related to public status.
- 2, See all articles regardless of their privacy.

Details of the request:

- URI: api/v8.1/article/{id}/{idPrivacity}/related/list
- Type: GET
- Required headings:
- Content-Type: application/json
- Authorization: TOKEN

Parameters:

Name	Data type	Obligatory	Description
Id	Number	Yes	Article ID
idPrivacity	Number	Yes	Item privacy id

Body of the petition:

Empty.

Answer:

The service will return a Json with the following structure:


```
[
 {
   "Description": "Ejemplo de descripción",
   "Id": 1,
   "Title": "Titulo del Artículo"
 }
]
```


Error messages:

Code	HTTP status	Error Message
400	BadRequest	InvalidArticleId
400	BadRequest	InvalidIdPrivacity
401	Unauthorized	InvalidToken
404	NotFound	InvalidArticleId
500	InternalServerError	FailureListRelatedArticles