

Interface specification for the transfer of legal texts from IT-Recht Kanzlei München to your system

Data is pushed to your interface via POST in XML format. The document is UTF-8 encoded. The POST parameter is “xml” by default (`$_POST['xml']`), other agreements are possible.

1 Implementation and use of the PHP SDK

We offer a PHP SDK for users of our IT-Recht Kanzlei interface. The SDK is designed for easy use. First, the two abstract methods of the class **LTIHandler** class and one of the two authentication methods must be overwritten.

1.1 Creation of your own LTI handler

1.1.1 Authentication methods

To ensure that the transfer of the legal texts originates from the IT-Recht Kanzlei system, you can implement one of the following two methods.

isTokenValid(string \$token): bool

The validity of the transmitted token must be checked here if the system should work with a token. The token is generated manually by you or automatically by your system and stored in the client portal of IT-Recht Kanzlei during configuration of the interface by the user of the interface. This is the preferred authentication method. To generate a random token see `LTI::generateToken()`.

validateUserPass(string \$username, string \$password): bool

The correctness of a transmitted user name and password must be checked here. The user name and password can be assigned by the users themselves in your plugin or can be created by your plugin. The user name and password must then be stored in the client portal when setting up the interface.

1.1.2 **handleActionGetAccountList(): LTIAccountListResult**

This function should return a list of all sales channels in your system. An ID (`accountid`) and the name (`accountname`) must be specified for each sales channel. In addition, a list of available target languages and countries can be transferred for each sales channel.

Even if your system is not a multishop system, it is recommended to implement this function to announce the supported target languages and countries of your system. For this use case, only one sales channel needs to be specified where the ID is specified as 0. The account name can remain empty.

Important: Please ensure that these 5 special XML characters are replaced by corresponding entities in the names of the sales channels (`accountname`):

```
& => &amp;  
< => &lt;  
> => &gt;  
" => &quot;  
' => &apos;
```

1.1.3 `handleActionPush(LTIPushData $data): LTIPushResult`

Once the request has been received by the plugin, the document must be integrated into the corresponding target page. The document is processed here. The `LTIPushData` object has getter methods for all properties of the document that can be queried and subsequently processed.

1.1.4 `preHandleRequest()`

This method can be used to initialize resources or to validate preconditions (eg. configuration settings) that the target system has to fulfill in order to operate.

If the necessary conditions are not met, you can throw an exception here which will be converted to a properly formatted error response.

1.2 Use of the `LTI` class and the `LTI` handler you have implemented.

The class `LTI` (`LTI.php`) takes over the preprocessing of the incoming XML data, evaluates the action to be executed and carries out most of the error handling. This file should not be changed to ensure that the SDK remains fully functional.

The constructor of the `LTI` class requires three parameters.

1. An instance of your overwritten `LTIHandler` class
2. The version of the system that is addressed with your implementation
3. The version of your implementation in which the SDK is used

The last step is to call the `handleRequest()` method of the `LTI` class object. For this purpose, `$_POST['xml']` can be used directly as a parameter.

Both error handling within the SDK and the preparation of the response for the IT-Recht Kanzlei server are handled automatically by the SDK. The developer can ignore these aspects.

An example implementation of the SDK is included in the supplied `example.php` file.

2 General

2.1 Versioning

Requirements for the processing of version numbers by the client portal of IT-Recht Kanzlei:

- `meta_modulversion`: Assign incrementing version numbers for your implementation when you first create it and for each subsequent update.
- `meta_shopversion`: If possible, please also transmit a system version number that can be compared using comparison operators (e.g. “2.0” instead of “XY2” - otherwise by request). Please let us know the structure of your versioning scheme (e.g. “major.minor”) so that we can process it correctly internally.

2.2 Multishop systems

If your system is a so-called multishop system, i.e. several sales channels/services exist under one administration interface, it is necessary that the user of your implementation is offered a list of available sales channels/services for which the legal texts are to be transferred in the client portal of the IT-Recht Kanzlei.

The list of sales channels is retrieved using the function `handleActionGetAccountList()`.

With the subsequent “push”, the `accountId` selected by the user is transmitted in the XML element `user_account_id` for multishop systems.

2.3 Best Practices

- Assign proper numerical, incrementing version numbers (e.g. “1.0”, “1.1”, “1.2”, ...) for your module when it is first created and for each subsequent update. In addition to classic version numbers, the release date can also be used numerically here (e.g. “20230827”, format YYYYMMDD). Always state the current version number in the documentation / installation instructions supplied with the module and at least in the main program file.
- Include easy-to-understand installation instructions with your module download or on the download page. Include special cases in your description (e.g. for older store versions).
- Include a description for the user of new versions of your implementation (updates) on how to update to the latest version (this often differs from the classic installation instructions), unless the update runs automatically (e.g. by clicking on “Update” in the store’s module store).
- Include a changelog.txt or similar with new releases of your implementation (updates) to inform the user about the new features and bug fixes.

3 Testing

To test your implementation of the interface, please read the README.md in the **testSuite** directory.

An additional tool you can use for testing is the LTI Test Tool. There you can enter the API URL and the authentication data for your test system and execute requests with the legal text interface of the IT-Recht Kanzlei.

4 Details of the implementation within the SDK

If you cannot or do not want to use the SDK, you will find some details below that could be helpful for your own implementation.

4.1 XML elements, [data type] and (possible values):

- `api_version` [string]
Version number of the interface (e.g. “1.0”)
 - `user_auth_token` [string]
Is generated by your system and can also be found there. Used for authentication.
 - `action` [string] (`push` | `getaccountlist` | `version`)
“`version`” only returns the store version, module version and PHP version installed on the system.
PHP version installed on the system.
“`getaccountlist`” lists all sales channels and their supported languages.
“`push`” is the command for transferring a legal text.
 - `user_account_id` [string]
is only set for multishop systems if a list of sales channels has previously been retrieved using the `getaccountlist` action and a selection has been made by the client.
-
- `rechtstext_type` [string] (`impressum` | `agb` | `datenschutz` | `widerruf`)
Type of legal text transferred
 - `impressum` = imprint
 - `agb` = general terms and conditions

- datenschutz = privacy policy
 - widerruf = cancellation policy
- rechtstext_title [string]
Title of the translated legal text in the original language
- rechtstext_text [text]
Text variant of the legal text
- rechtstext_pdf [text]
PDF version of the legal text
- rechtstext_html [text]
HTML version of the legal text
- rechtstext_country [string]
ISO 3166-1-alpha-2, country, e.g. “DE” for Germany, is transmitted uppercase
- rechtstext_language [string]
ISO 639-1, language of the legal text, e.g. “de” for German, is transmitted lowercase
- rechtstext_language_iso639_2b [string]
ISO 639-2 bibliographic code (B code), language of the legal text, e.g. “ger” for German, lowercase

4.2 Status codes

- **success:** Everything went well. You confirm,
 - that no errors occurred when querying the version
 - that no errors occurred when querying the account list
 - that when a legal text was pushed, it was successfully published in the user’s account/shop
- **error:** Regardless of the error code, the status of the error response will always be “error”.

4.3 Error codes

The error codes are documented in the LTIError.php file. If possible, please only use the error codes defined in the file. Avoid using the error code 99 as much as possible. You can define your own error codes with the number range ≥ 100 . Please inform IT-Recht Kanzlei of the error code and its meaning. Error codes for other generic errors can also be added to the number range < 100 by agreement.

4.4 Example XML response output

4.4.1 “version” if successful:

```
<?xml version="1.0" ?>
<response>
  <status>success</status>
  <meta_shopversion>1.0</meta_shopversion>
  <meta_modulversion>1.1.0</meta_modulversion>
  <meta_phpversion>7.4</meta_phpversion>
</response>
```

4.4.2 “getaccountlist” if successful:

```
<?xml version="1.0" ?>
<response>
  <status>success</status>
  <meta_shopversion>1.0</meta_shopversion>
  <meta_modulversion>1.1.0</meta_modulversion>
  <meta_phpversion>7.4</meta_phpversion>
  <account>
    <accountid>12345</accountid>
    <accountname>Shop 1</accountname>
    <locales>
      <locale>de</locale>
      <locale>en</locale>
      <locale>fr</locale>
    </locales>
    <countries>
      <country>DE</country>
      <country>AT</country>
      <country>GB</country>
      <country>FR</country>
    </countries>
  </account>
  <account>
    <accountid>23456</accountid>
    <accountname>Shop 2</accountname>
    <locales>
      <locale>de</locale>
    </locales>
    <countries>
      <country>DE</country>
    </countries>
  </account>
</response>
```

4.4.3 “push” if successful:

```
<?xml version="1.0" ?>
<response>
  <status>success</status>
  <meta_shopversion>1.0</meta_shopversion>
  <meta_modulversion>1.1.0</meta_modulversion>
  <meta_phpversion>7.4</meta_phpversion>
  <target_url>https://example.org/imprint</target_url>
</response>
```

4.4.4 In the event of an error:

```
<?xml version="1.0" ?>
<response>
  <status>error</status>
  <meta_shopversion>1.0</meta_shopversion>
  <meta_modulversion>1.1.0</meta_modulversion>
  <meta_phpversion>7.4</meta_phpversion>
  <error>12</error>
  <error_message>Something was not found!</error_message>
</response>
```

5 Contact

Herr Max-Lion Keller, LL.M.
IT-Recht Kanzlei, Alter Messeplatz 2, 80339 München
Phone: +49 89 1301433-0
Fax: +49 89 1301433-60
E-mail: m.keller@it-recht-kanzlei.de