Schnittstellen-Spezifikation zur Übertragung von Rechtstexten der IT-Recht Kanzlei München zu Ihren Systemen

Version: V1.0.0 – Stand: 01. Dezember 2021

Daten werden per POST im XML-Format an Ihre Schnittstelle gepusht. Als Encoding ist UTF-8
vorgesehen. Der POST-Parameter lautet standardmäßig "xml“ ($_POST['xml']), andere Vereinbarungen sind möglich.

1. XML-Elemente, [Datentyp] und (mögliche Werte):

- api_version [string]
  Versionsnummer der Kommunikationsschnittstelle (z.B. „1.0“)
  ……...
- user_auth_token [string]
  Wird vom Shopsystem generiert und ist auch dort zu entnehmen. Dient der Authentifizierung bei einem POST-Request.
- user_account_id [string]
  wird nur für Multishop-Systeme gesetzt , wenn zuvor durch action „getaccountlist“ eine Shop-Liste abgerufen und vom Mandanten eine Auswahl getroffen wurde
  ……...
- rechtstext_type [string] (agb | datenschutz | widerruf | impressum)
  Art des übertragenen Rechtstextes
- rechtstext_title [string]
  Titel des übertragenen Rechtstextes in Originalsprache
- rechtstext_text [text]
  Text-Variante des Rechtstextes
- rechtstext_pdf [text]
  PDF -Variante des Rechtstextes
- rechtstext_html [text]
  HTML-Variante des Rechtstextes
- rechtstext_country [string]
  ISO 3166-1-alpha-2, Land, z.B. „DE“ für Deutschland, wird uppercase übermittelt
- rechtstext_language [string]
  ISO 639-1, Sprache d. Rechtstextes, z.B. „de“ für Deutsch, wird lowercase übermittelt
- rechtstext_language_iso639_2b [string]
  ISO 639-2 bibliographic code (B code), Sprache d. Rechtstextes, z.B. „ger“ für Deutsch, lowercase
  ……...
- action [string] (push | getaccountlist | version)
  „push“ ist der Standardbefehl für eine Übertragung. „getaccountlist“ ist nur für Mutlishop-Systeme vorgesehen, d.h. die unter einem user_auth_token mehrere Shops betreiben. “version” gibt lediglich die Shop-Version, Modul-Version und auf dem System installierte PHP-Version zurück.

2. Als Stati /Errorcodes Ihrer Schnittstelle sind folgende Werte vorgesehen:

Stati

- success
  Es hat alles geklappt. Sie bestätigen, dass der Rechtstext erfolgreich im Account/Shop des Users publiziert wurde
- verison
  Enthalten in der erfolgreichen Response wenn die Versionen des Moduls angefragt wurden
- error
  Unabhängig vom Fehlercode wird der Status der Fehler-Response immer “error” sein

Errorcodes

- error 1
  Schnittstellen-Version
- error 3
  Fehler beim Authentifizieren des Users, d.h. user_auth_token nicht korrekt
- error 4
  Wert für rechtstext_type ist leer oder gesendeter Typ wird nicht unterstützt
- error 5
  Wert für rechtstext_text ist leer
- error 6
  Wert für rechtstext_html ist leer
- error 9
  Wert für rechtstext_language ist leer
- error 10
  Wert für action ist leer
- error 11
  Wert für user_account_id wird benötigt (Multishop-System), ist aber leer
- error 12
  Fehler beim Verarbeiten der XML-POST-Daten
- error 17
  Wert für rechtstext_country ist leer
- error 18
  Wert für rechtstext_title ist leer
- error 80
  Die Schnittstellenkonfiguration auf Shopseite wurde noch nicht vollständig vom Nutzer abgeschlossen (Beispiele: Rechtstexteseiten aus CMS noch nicht manuell zugeordnet, manuelle Generierung eines Auth-Tokens noch nicht erfolgt)
- error 81
  Die CMS-/Textseite im Shop, in die der Rechtstext abgelegt werden soll, wurde nicht gefunden.
- error 99
  sonstiger/nicht näher spezifizierter Fehler (Sammelcode für alle anderen Fehler)

3. Implementierung PHP-SDK

Für die Nutzer unserer IT-Recht Kanzlei Schnittstelle mit PHP Backend bieten wir ein überarbeitetes SDK an. Das SDK ist sehr übersichtlich gestaltet und leicht zu verwenden. Im wesentlichen müssen dazu lediglich drei Punkte bearbeitet werde.
Zunächst müssen die drei abstrakten Methoden der LTIHandler-Klasse überschrieben werden.

1. isTokenValid(string $token)

Hier muss die Korrektheit des hereinkommenden Tokens geprüft werden.

2. handleActionPush(LTIPushData $data)

Nachdem der Request am Plugin angekommen ist, muss das Dokument in die entsprechende Zielseite integriert werden. Die Integration des Dokuments wird hier vorgenommen. Das $data Objekt besitzt Getter-Methoden für alle wichtigen Informationen bezüglich des Dokuments die so abgefragt und anschließend weiterverarbeitet werden können.

3. handleActionGetAccountList()

Handelt es sich beim Zielsystem um ein Multishop-System, kann so eine Liste aller im Zielsystem vorhandenen Shops abgerufen werden. Um diese Methode nutzen zu können, muss bei beim Erzeugen des Objekts der LTI-Klasse der letzte Parameter unbedingt auf true gesetzt werden. Andernfalls wird ein Fehler zugegeben, dass das System kein Multishop-System ist.

Anschließend muss ein Objekt der LTI-Klasse des SDKs erzeugt werde. Der Konstruktor benötigt vier Parameter.

1. Eine Instanz der überschriebenen LTIHandler-Klasse
2. Die Version des Shops, der mit diesem Plugin angesprochen wird
3. Die Version des Plugins in dem das SDK verwendet wird
4. Ein Boolean der angibt, ob das Zielsystem ein Multishop-System ist oder nicht

Als letzter Schritt muss die **handleRequest**-Methode de LTI-Klassen Objekts aufgerufen werden. Hierzu kann **$_POST** direkt als Parameter genutzt werden.

Das SDK behinhaltet eine LTI.php Datei, in welcher die LTI-Klasse implementiert ist. Diese übernimmt die Koordination der hereinkommenden Post-Daten, wertet die auszuführende Aktion aus und führt einen Großteil der Fehlerbehandlung durch. Außerdem inkludiert sie alle weiteren im SDK enthaltenen und benötigten Datein. Um die Lauffährigkeit des SDKs zu gewährleisten sollte diese Datei nicht geändert werden.

Sowohl Fehlerbehandlung innerhalb des SDKs, als auch das Senden der Response zum IT-Recht Kanzlei Server werden vom SDK automatisch behandelt. Der Entwickler kann diese Aspekte ignorieren.

Nachfolgend wird eine Beispielimplementierung des SDK gezeigt.

```php
require_once __DIR__ . "/src/LTI.php";

class MyLTIHandler extends \ITRechtKanzlei\LTIHandler {
  public function isTokenValid(string $token): bool {
    // Validate your token here
    return $token == '12345678';
  }

  public function handleActionPush(\ITRechtKanzlei\LTIPushData $data):\ITRechtKanzlei\LTIPushResult {
    // Implement the logic to store your pushed document to the shop here and return an
        object of ITRechtKanzlei\LTIPushResult with
    // an url where to find the currently uploaded document. Replace the url with your document url.
    return new \ITRechtKanzlei\LTIPushResult('https://www.examplepage.com');
  }

  public function handleActionGetAccountList(): \ITRechtKanzlei\LTIGetAccountListResult {
    // add all your shops here to the $accoutlist like seen in the example.
    $accountList = new \ITRechtKanzlei\LTIGetAccountListResult();
    $accountList->addAccount('1', 'example store name 1');
    $accountList->addAccount('2', 'example store name 2');
    $accountList->addAccount('3', 'example store name 3');

    return $accountList;
  }
}


//  Instantiate object of class that overrides abstract methods
$ltiHandler = new MyLTIHandler();


//  Instantiate of ITRechtKanzlei\LTI and call handleRequest(...) method
$lti = new \ITRechtKanzlei\LTI($ltiHandler, '1.2', '1.0', true);
$lti->handleRequest($_POST);
```

4. Allgemeines

Guidelines für Rechtstexte-Schnittstellen als downloadbares Modul

- Vergeben Sie für Ihr Modul bei Erstersterstellung und bei jedem folgenden Update ordentliche numerische, hochzählende **Versionsnummern** (z.B. "1.0", "1.1", "1.2", ...). Neben klassischen Versionsnummern kann hier auch numerisch das Veröffentlichungsdatum genutzt werden (z.B. „20140707“, Format YYYYMMDD). Nennen Sie die aktuelle Versionsnummer immer in der dem Modul beiliegenden Dokumentation / Installationsanleitung und mindestens in der Haupt-Programmdatei.
- Fügen Sie Ihrem Modul-Download oder auf der Download-Seite eine gut verständliche **Installationsanleitung** bei. Berücksichtigen Sie in Ihrer Beschreibung auch Sonderfälle (z.B. für ältere Shopversionen).
- Fügen Sie aktualisierten Modul-Versionen (Updates) eine Beschreibung für den Nutzer bei, wie die **Aktualisierung** auf die neueste Modulversion durchzuführen ist (diese weicht oft von der klassischen Installationsanleitung ab), sofern das Update nicht automatisch abläuft (z.B. durch einen Klick auf „Update“ im Modul-Store des Shops).
- Fügen Sie aktualisierten Modul-Versionen (Updates) eine **changelog.txt** o.ä. bei, um den Nutzer über die Neuerungen zu informieren.

Versionierung

Voraussetzungen für die Verarbeitung der Versionsnummern durch das Mandantenportal der IT-Kanzlei:

- <meta_modulversion>: Vergeben Sie für Ihr Modul bei Erstersterstellung und bei jedem folgenden Update hochzählende Versionsnummern, die Sie in diesem XML- Element übermitteln.
- <meta_shopversion>: Hier wenn möglich ebenfalls eine durch Vergleichsoperatorenvergleichbare Shopversionsnummer übermitteln (z.B. "2.0" statt "GX2" - andernfalls nach Absprache). Teilen Sie uns bitte die Struktur Ihrer Versionierung mit (z.B. „major.minor“), damit wir diese intern korrekt verarbeiten.

Multishop-Systeme

Falls es sich bei Ihrem System um ein sog. Multishop-System handelt, d.h. dass unter einem user_auth_token mehrere Shops/Dienste existieren, ist es erforderlich, dass dem User auf der IT-Recht-Kanzlei - Seite zunächst ein Dialog gezeigt wird, in welchem er auswählt, in welche Shops/Dienste der neue Rechtstext gepusht werden soll.

Dazu muss auf Ihrer Seite der Schnittstelle zusätzlich die Funktion "action = getaccountlist" implementiert werden, welche zunächst einen Abruf der User-Shops/Dienste ermöglicht und im Ergebnis eine Liste in folgendem XML-Format liefert.

Wichtig: Bitte achten Sie bei den Shopnamen (<accountname>) darauf, dass die (5) XML-Sonderzeichen durch entsprechende Entities ersetzt werden, also

& = & < = < > = > " = " ' = '

Bei dem anschließenden call "action = push" wird bei Multishop-Systemen die vom User ausgewählte accountid im XML-Element "user_account_id" mit übermittelt.

5. Beispiel XML-Response Ausgaben

(graue Angaben sind Beispiel Werte)

Bei Erfolg “push”:

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

Bei Erfolg “getaccountlist”:

```
<?xml version="1.0" ?>
<response>
	<accountlist>
		<account>
			<accountid>12345</accountid>
			<accountname>Shop 1</accountname>
		</account>
		<account>
			<accountid>23456</accountid>
			<accountname>Shop 2</accountname>
		</account>
	</accountlist>
	<meta_shopversion>1.0</meta_shopversion>
	<meta_modulversion>1.1.0</meta_modulversion>
	<meta_phpversion>7.4</meta_phpversion>
</response>
```

Bei Erfolg “version”:

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

Bei einem Fehler:

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

6. Best Practices

- Vergeben Sie für Ihr Modul bei Erstersterstellung und bei jedem folgenden Update ordentliche numerische, hochzählende Versionsnummern (z.B. "1.0", "1.1", "1.2", ...). Neben klassischen Versionsnummern kann hier auch numerisch das Veröffentlichungsdatum genutzt werden (z.B. „20140707“, Format YYYYMMDD). Nennen Sie die aktuelle Versionsnummer immer in der dem Modul beiliegenden Dokumentation / Installationsanleitung und mindestens in der Haupt-Programmdatei.
- Fügen Sie Ihrem Modul-Download oder auf der Download-Seite eine gut verständliche Installationsanleitung bei. Berücksichtigen Sie in Ihrer Beschreibung auch Sonderfälle (z.B. für ältere Shopversionen).
- Fügen Sie aktualisierten Modul-Versionen (Updates) eine Beschreibung für den Nutzer bei, wie die Aktualisierung auf die neueste Modulversion durchzuführen ist (diese weicht oft von der klassischen Installationsanleitung ab), sofern das Update nicht automatisch abläuft (z.B. durch einen Klick auf „Update“ im Modul-Store des Shops).
- Fügen Sie aktualisierten Modul-Versionen (Updates) eine changelog.txt o.ä. bei, um den Nutzer über die Neuerungen zu informieren.

7. Kontakt

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