Dialogue SOAPPortal

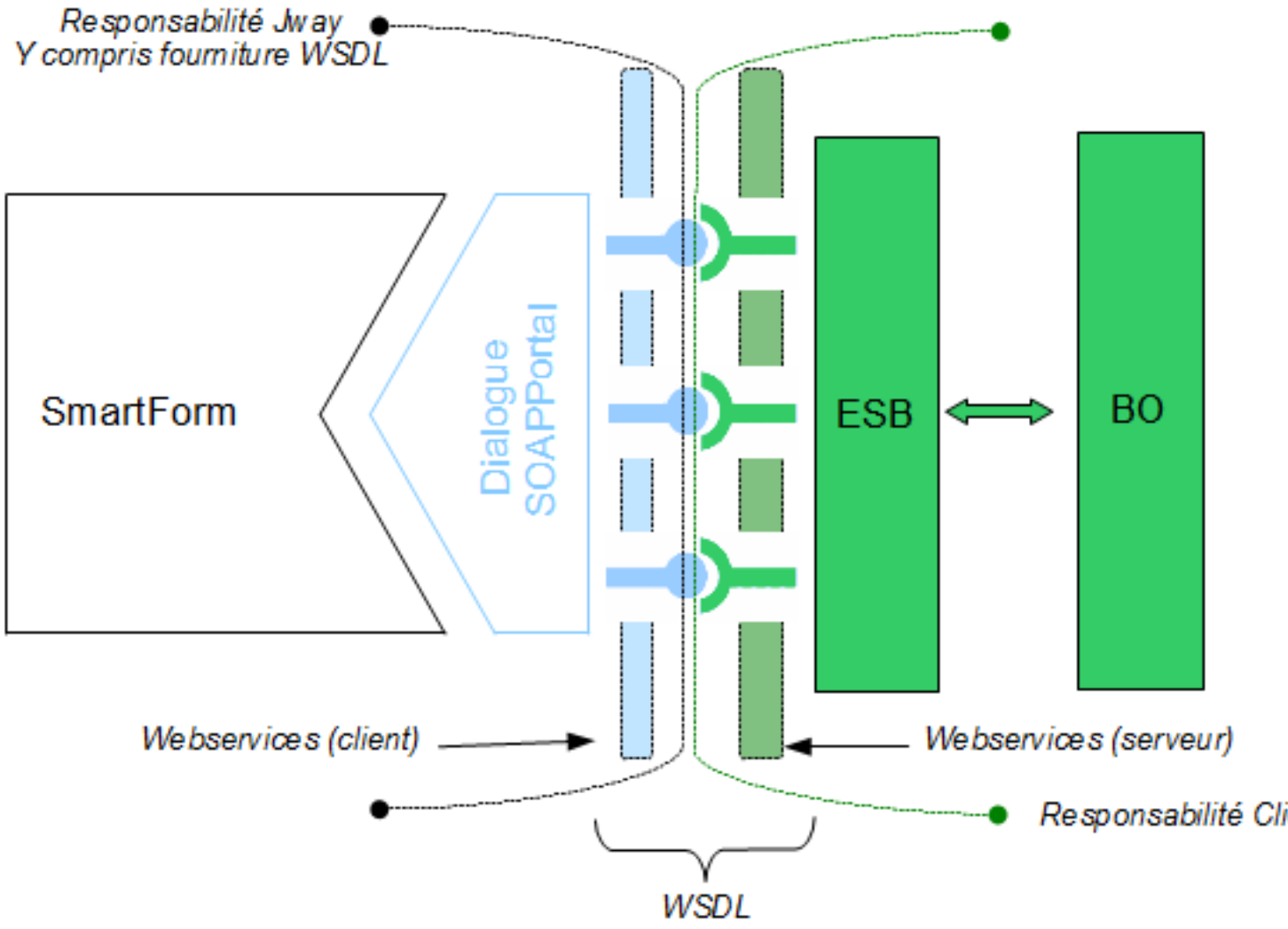
FormPublisher intègre la notion de dialogue applicatif qui permet d'échanger de façon automatique des données avec un environnement durant les différents phases de vie d'un formulaire.

Comme tout dialogue applicatif, celui-ci découle d'une phase d'intégration visant à mettre en relation les SmartForms et un Back Office éventuellement à travers un ESB.

Les données échangées entre les deux parties sont contractualisées à travers un fichier de type Web Service Definition Language (WSDL) construit d'une manière assez générique pour permettre de prendre en compte des variantes dans les données échangées.

Ce document décrit les éléments intervenants dans la mise en oeuvre de ce dialogue, des éléments de sécurités et les données échangées.

1. Présentation



SOAPPortal est un dialogue applicatif permettant d'échanger des données entre une web application produite avec FormPublisher V2.1 (version communication pack) et un ensemble de services de type SOAP/DOCUMENT/LITTERAL.

Ce dialogue prend place à l'initiative d'un *SmartForm* suite à son démarrage en mode SOAPPortal, et ce lors des phases de vie d'un formulaire.

Au mode **SOAPPortal** est associé un WSDL qui décrit les attentes de FormPublisher en terme de services à offrir de la part d'un serveur.

Ce WSDL décrit en faite plusieurs catégories de services pouvant être exploitées directement par FormPublisher ou des applications complémentaires (serveur de gestion des annexes, serveur de signature, …) .

Une version **actualisée** de ce WSDL peut être obtenue auprès de JWay sur simple demande.

Les opérations **exploitées** directement par FormPublisher :

initializeForm

Ce service peut être appelé par FormPublisher pour générer un numéro identifiant la session de travail. Ce sera par exemple le cas si le formulaire ne possède pas de numéro identifiant la session lors de l'appel aux services saveForm, validateForm ou exitForm.

openForm

Ce service est appelé lors du lancement d'un formulaire si celui-ci doit préalablement demander des données issues d'une précédente session par exemple.

saveForm

Ce service est appelé lors de l'action de sauvegarde du formulaire. Les données envoyées le sont en fonction du paramétrage de l'action de sauvegarde.

validateForm

Ce service est appelé lors de l'action de validation du formulaire. Les données envoyées dépendent du paramétrage du formulaire. Il y aura production d'un xml de sortie ainsi qu'éventuellement un document bureautique (pdf par exemple).

exitForm

Ce service est appelé lors de l'action de sortie du formulaire. Les données envoyées le sont en fonction du paramétrage de l'action de sortie.

Comme l'indique le schéma, l'usage du dialogue applicatif SOAPPortal implique un composant serveur. La réalisation de ce serveur est à la charge du client qui peut se baser sur le WSDL pour en générer les services avec des outils de développement traditionnels. JWay peut éventuellement offrir un accompagnement dans cette réalisation.

Par ailleurs, pour qu'un SmartForm fonctionne en dialogue applicatif SOAPPortal, il faut que celui-ci soit invoquée dans ce mode. L'invocation nécessite l'usage d'une bibloithèque fournie par JWay pour crypter des paramètres sensibles au sein de l'URL d'appel du SmartForm.

2. Invocation d'un SmartForm en mode SOAPPortal

A l'aide de la bibliothèque de FormPublisher, nous pouvons générer une URL permettant d'invoquer un smartForm en dialogue applicatif SOAPPortal :

Les paramètres en **vert** apparaissent directement dans l'URL.

Les paramètres en **rouge** devienne un paramètre unique crypté en fonction de la clé **useKey**.

L'invocation en mode dialogue applicatif SOAPPortal se traduit par la présence du paraèmtre dialog=SOAPPortal.

Le paramètre jsysPORTALRETRIEVE indique s'il faut ou non faire appel au webservice openForm au lancement du formulaire.

userId et draftId représentent des identifiants pour un utilisateur et une demande. Il faut toujours affecter une valeur au userId qui sera ou non traitée par le serveur. Pour le draftId cette valeur peut être obtenue en dehors de la phase de démarrage du formulaire.

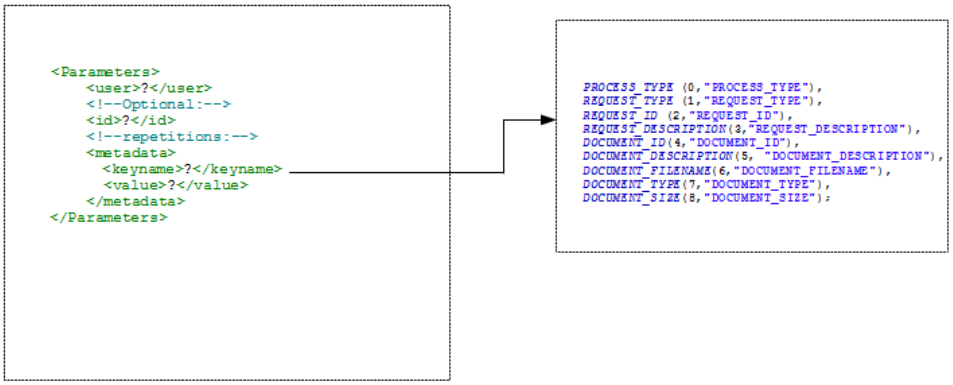
```
SOAPPortalURL spurl=new SOAPPortalURL();

 spurl.setServer("http://smartform.jway.lu");
 spurl.setWebAppName("Test");
 spurl.setAction("login");
 spurl.setLayout("normal");
 spurl.setMediaType("ji_html");
 spurl.setDocumentId("form");
 spurl.setJsysPORTALUSERID(userId);
 spurl.setJsysPORTALDRAFTID(draftId);
 spurl.setJsysPORTALRETRIEVE("yes".equalsIgnoreCase(retrieve));
 if(isNEMPTY(metadata)){
      spurl.setJsysMETADATA(metadata);
 }

 spurl.setUseKey(useKey);

 result=spurl.getURL();
```

```
http://smartform.jway.lu/Test/Controler?
action=login&documentId=form&ids=userData&ids=draftData&sds=draft
Data&mediaType=ji_html&dialogAUTH=5ff7666da9148abd1f200e6d8824685
72d249a28ba8f4e96d8a8718082bed911835af4869ad7d3e67be57230f42761e3
f8dc7050471e71a1567baca3b0c62b8a&dialog=SOAPPortal
```

3. Données émises pour l''appel des webservices

Nous allons présenter les fluxs XML utilisés lors des échanges avec le serveur implémentant le WSDL.

Ici la structure telle que produite en analysant le WSDL dans le cas "simple".



Nota : Cette structure est mise en œuvre au moment de l'appel de **initializeForm** ou de **openForm**.

Les metadata doivent avoir un **keyname** ayant une valeur parmi la liste en bleu (PROCESS_TYPE , ...) .

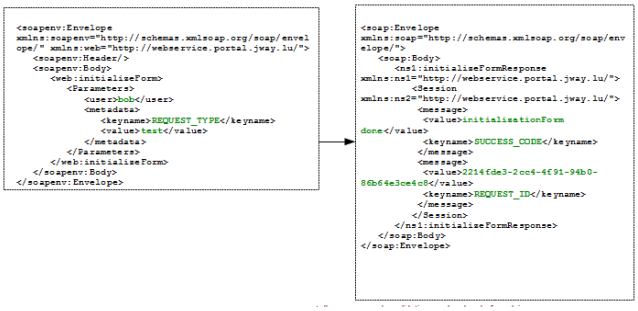
Le champs obligatoire lors de ces appels est **user**.

4. Service initializeForm

Le schéma ci dessous présente à gauche la structure émise lors de l'appel du webService et à droite la structure reçue en réponse.

L'appel d'un webService se décompose en général en paramètres attendus (comme *<user>*) ou en *metadata*.

La réponse est donnée sous la forme d'un ensemble de *messages* contenus dans une *session*.



Les données attendues dans la réponse sont dépendantes du service appelé mais doivent toujours inclure un message indiquant si la réponse est positive ou négative. Dans le cas de initializeForm, il est également attendu un message contenant le REQUEST_ID produit pour identifier le dossier.

L'appel à initializeForm peut avoir lieu au démarrage du formulaire ou au moment d'une sauvegarde, validation ou abandon du formulaire en cours.

Ce comportement est dirigé par la variable d'environnement : jway.SOAPPortal.id.creation.optimistic

Par défaut cette valeur est à true, c'est à dire que l'obtention d'un draftId peut avoir lieu au moment de la sauvegarde/validation/exit s'il n'est pas encore attribué.

5. Service openForm

Le demande openForm se traduit par la présence de user et de numéro identifiant la demande à laquelle on souhaite accéder.

La réponse doit contentir son status (ici SUCCESS_CODE) ainsi que les éléments retournés sous forme de dataStores userData et/ou formData. A noter que ces dataStores peuvent être itératifs.

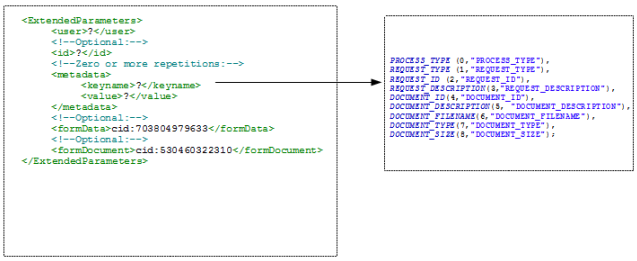
Dans ce cas le serveur a répondu avec un traitement multipart des données encodées

6. Données émises pour l'appel des webservices complexes.

Certains des webservices utilisés par le dialogue applicatif SOAPPortal demandent des échanges de données plus complexes, comme dans le cas de saveForm, validateForm et exitForm.

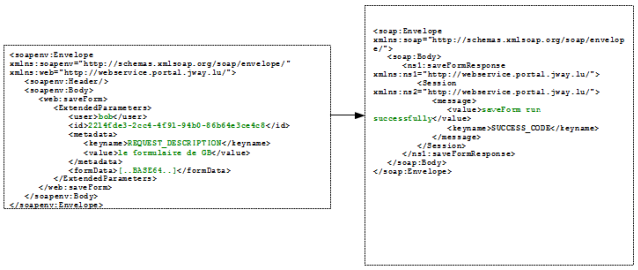
Cette structure est représentée ci-après.

Elle conserve les paramètres user, id et metadata mais rajoute les formData et FormDocument .



7. Service saveForm

Cet échange a lieu lorsque l'usager du formulaire demande une sauvegarde de celui-ci en cours de remplissage.



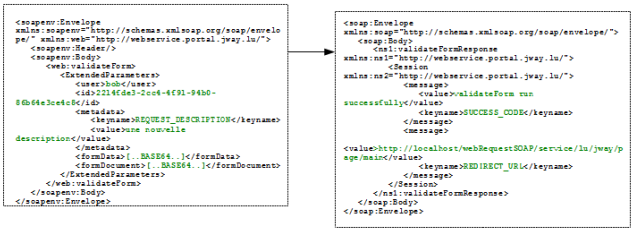
Cette structure est utilisée lors de l'appel du service saveForm.

Les données issues du formulaire , le datastore, est communiquée sous forme d'un XML encodé en BASE64 au service dans le paramètre formData, ainsi que l'utilisateur et l'identifiant de la demande.

Un metadata de keyname REQUEST_DESCRIPTION est également communiqué. Celui-ci est issu de champs de saisie du formulaire.

8. Service validateForm

Ce service est appelé lorsque l'usager du formulaire a terminé son remplissage et souhaite en soumettre le contenu.

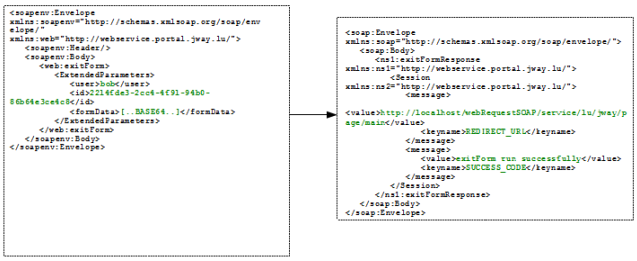


Dans les données communiquées, il y aura un dataStore (formData) ainsi qu'un potentiellement un document (formDocument) de type définit lors de la validation ainsi que les informations habituelles comme dans saveForm.

Le serveur a en plus du code de retour spécifié une URL de redirection vers laquelle le browser va rediriger l'usager.

9. Service exitForm

Ce service est appelé lors de l'opération d'abandon de remplissage du formulaire.



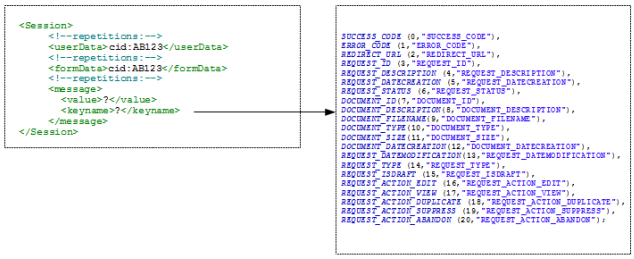
Durant cette opération, le service exitForm est contacté et les données du formulaire lui sont communiquées de sorte à pouvoir être éventuellement conservées ou analysées.

Dans le cas présent, le service indique que l'opération c'est bien déroulée et demande une redirection.

10. Structure XML retournée.

Nous avons vu les différents services exploités par SOAPPortal ainsi que les structures de données échangées en appel et en réponse.

Le schéma ci après présente la forme que peut prendre une réponse ainsi que les messages possibles



La session comprend toujours au moins un message indiquant le résultat de l'appel au web service, à savoir SUCCESS_CODE ou ERROR_CODE, plus d'autres éléments éventuellement.

11. Paramétrages d'un SmartForm en mode SOAPPortal

Il est possible de mettre en place des propriétés de publication pour influer sur le fonctionnement d'un smartForm en mode SOAPPortal :

Les appels des webservices peuvent passer par un proxy :

jway.portal.proxy.isSet=true

jway.portal.proxy.host=localhost

jway.portal.proxy.port=9900

La localisation du serveur de web services :

jway.SOAPPortal.transport=http://localhost:8888/SOAPPORTALServerCXFSPRING/PortalServer

La clé d'encryptage :

jway.SOAPPortal.encryption.key=*********

Obtention d'un numéro de demande si celui-ci n'est pas encore attribué :

jway.SOAPPortal.id.creation.optimistic=false

Type du datastore à produire lors des actions:

jway.SOAPPortal.logout.join.ds=complete

jway.SOAPPortal.save.join.ds=complete

jway.SOAPPortal.valid.join.ds=strict

Activation d'actions customisées :

jway.SOAPPortal.action.extension.list=logout,submit

12. Sécurisation des échanges en mode SOAPPortal

Certains web applications voient leur l'accès limité par l'absence ou la présence d'éléments posés directement au niveau des headers de la requête HTTP.

Si une telle web application est produite par FormPublisher, il peut être nécessaire d'exploitées ces informations.

FormPublisher de façon native récupère ces informations et sait les communiquer aux autres éléments motorisant un formulaire. L'exploitation de ces données restent alors à mettre en place.

Dans le contexte du dialogue SOAPPortal, cette approche va plus loin en permettant le passage des ces informations jusqu'aux appels de webservice.

Les variables extraites se retrouvent dans le record jsysSEC et sont donc accessibles. A noter que si l'auteur modélise un webService qui exploite ce genre de sécurité, ce sera à sa charge d'intégrer ce code dans le fichier .properties associés.

Par défaut cette modification a été apportée pour tous les échanges de type SOAPPortal.

Les variables extraites se retrouvent dans le record jsysSEC et sont donc accessibles. A noter que si l'auteur modélise un webService qui exploite ce genre de sécurité, ce sera à sa charge d'intégrer ce code dans le fichier .properties associés.

Par défaut cette modification a été apportée pour tous les échanges de type SOAPPortal.

Pour ce faire, il faut créer dans son APPLI.REF une extension de signature :

package lu/jway/webapp/dialog et nom de la classe HTTPHEADERSFilter.java ;

cette classe doit implémenter l'interface : lu.jway.webapp.dialog.IHTTPHEADERSFilter

Cette classe va avoir pour objectif de ne retenir que les headers nécessaires et éventuellement leur affecter un nom compatible avec le système de nommage de variables utilisés dans FormPublihser.

Ces valeurs seront intégrés dans le contexte du dialogue SOAPPortal et seront également visibles en tant que données d'instanciation de l'espace de données userData.

L'exploitation au niveau d'un template de properties peut se faire de la sorte :

```
<?if (existsAndNotEmpty($(jsysSEC)))?>
  <?foreach j in $(jsysSEC)?>
      <?xml-data "<entry key='"+$(j|key)+"'>"?>
      <?data $(j|value)?>
      <?xml-data "</entry>"?>
 <?end-foreach?>
<?end-if?>
```

Les variables extraites se retrouvent dans le record jsysSEC et sont donc accessibles. A noter que si l'auteur modélise un webService qui exploite ce genre de sécurité, ce sera à sa charge d'intégrer ce code dans le fichier .properties associés.

Par défaut cette modification a été apportée pour tous les échanges de type SOAPPortal.

13. Sécurisation avancée des échanges

Certains webServices demandent une sécurisation encore plus complexes, qui peut avoir une incidence directe sur les contenus échangés comme cela pourrait être le cas si ces contenus sont encryptés.

Ceci va plus loin que le fait d'injecter des nouveaux headers dans une requête HTTP.

Le moteur de webServices de FormPublisher intègre une telle proche qui s'adapte à l'usage de librairie comme WSS4J.

Pour influer sur des webServices de la sorte, il faut :