Unica Plan 12.1 - Module d'intégration

Table des matières

Chapitre 1. Présentation d'Unica Plan Integration Services

Unica Plan Unica Plan Integration Services combine les services Web Integration, les procédures d'API SOAP et les déclencheurs pour augmenter les fonctionnalités métier.

Unica Plan Integration Services se compose des éléments suivants :

- **Unica Plan Service Web Integration**

  La fonction Integration Services permet aux clients Unica Plan, ainsi qu'aux applications Professional Services d'intégrer Unica Plan à d'autres applications qui s'exécutent dans leur environnement.
- **Unica Plan Procédures et API SOAP de**

  Vous pouvez définir des procédures personnalisées dans Unica Plan pour étendre la logique métier Unica Plan de façon arbitraire. Une fois définies, ces procédures peuvent être les cibles des appels de service Web Integration Services provenant d'autres applications. Il est également possible de définir des procédures pour envoyer des messages à d'autres applications.
- **Unica Plandéclencheurs**

  Les déclencheurs peuvent être associés à des événement et à des procédures dans Unica Plan. Lorsque ce type d'événement se produit, le déclencheur associé est exécuté.

Les API REST n'utilisent pas les services d'intégration de Unica Plan. Pour des informations sur l'API REST, voir le guide d'administration d' Unica Plan.

Gestion des versions et compatibilité amont

Les futures versions des services d'intégration seront compatibles avec les versions antérieures et toutes les éditions secondaires et de maintenance qui partagent le même numéro de version. Toutefois, se réserve le droit d'abandonner la compatibilité avec la version antérieure pour les éditions principales "point zéro" (x.0) si le script commercial ou technique le justifie.

Le numéro de version principal de cette API est incrémenté si l'une des modifications suivantes est apportée :

- modification de l'interprétation des données ;
- modification de la logique métier (par exemple, modification des fonctions de méthode de service) ;
- modification des paramètres de méthode et/ou des types de retour.

Le numéro d'édition de cette API est incrémenté si l'une des modifications ci-après est apportée. Ces modifications sont compatibles avec une version antérieure par définition.

- ajout d'une nouvelle méthode ;
- ajout d'un nouveau type de données et restriction de son utilisation à une nouvelle méthode ;
- ajout d'un nouvel élément à un type énuméré ;
- définition d'une nouvelle version d'interface avec un suffixe de version.

Authentification

L'authentification n'est pas requise ; tous les clients sont associés à un utilisateur Unica Plan connu appelé PlanAPIUser. Les fonctions de sécurité de cet utilisateur spécial sont configurées par un administrateur système en fonction des besoins de tous les clients du service Web.

Environnement local

La seule langue prise en charge est la langue actuellement configurée pour l'instance de système Unica Plan . Les données qui dépendent de l'environnement local, tel que les messages et la devise, sont en principe intégrées à cet environnement.

Gestion d'état

Les API et le service Web sont sans état ; aucune information par client n'est sauvegardée par la mise en oeuvre du service au fil des appels API. Cette fonction permet de bénéficier d'une mise en œuvre de service plus efficace et simplifie la prise en charge du cluster.

Transactions de base de données

Unica Plan Integration Services ne montre pas les transactions de base de données au client, mais utilise ces informations si elles sont incluses dans le contexte d'exécution. Si une transaction est démarrée, l'effet de tous les appels API au sein d'une procédure particulière peut être atomique. Cela signifie qu'un échec d'appel API laisse la base de données dans le même état que si aucun appel API n'avait été émis. Les autres utilisateurs de Unica Plan ne voient pas les modifications tant que la procédure n'a pas validé la transaction.

Les appels API qui mettent à jour la base de données doivent tout d'abord acquérir un verrou d'édition afin d'empêcher les autres utilisateurs API de modifier les données sous-jacentes durant les appels API. Les autres utilisateurs ne peuvent pas mettre à jour les composants verrouillés tant que l'appel API n'est pas terminé. De même, l'utilisateur ou le client API Unica Plan suivant doit acquérir le verrou des données avant la soumission d'un autre appel API.

Traitement de l'événement

Les opérations effectuées sur les composants Unica Plan Unica Plan via cette API génèrent les mêmes événements que si l'opération avait été effectuée par un utilisateur Web de . Les utilisateurs qui ont souscrit à certaines notifications (par exemple, un changement d'état d'un projet) seront avertis des changements d'état résultant des appels API et des actions des utilisateurs.

Conditions requises pour Unica Plan Integration Services

Les conditions suivantes sont requises pour Unica Plan Integration Services.

La fonction Unica Plan Integration Services doit :

- Coupler de façon souple l'intégration du système
- Fournir un mécanisme permettant aux applications du client d'affecter Unica Plan via des appels de service Web

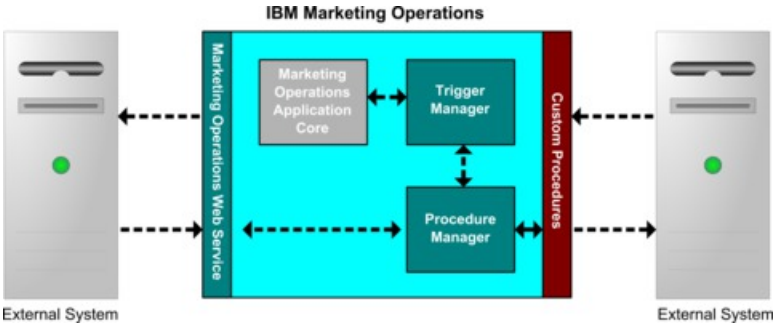
- Fournir un mécanisme permettant aux applications du client d'être averties de certains événements dans Unica Plan
- Fournir un modèle de programmation simple, facile à comprendre et à utiliser
- Etre robuste en cas de reprise sur incident
- Garantir l'intégrité des données
- S'intégrer aux clients Unica Plan basés sur l'interface graphique existants et réduire les effets sur ces derniers
- Fournir un accès à granularité fine aux composants Unica Plan tout en isolant les programmateurs des détails de mise en oeuvre sous-jacents

Généralités sur Unica Plan Integration Services

Unica Plan Integration Services permet de créer des procédures personnalisées. Vous pouvez utiliser ces procédures pour déclencher des événements externes lorsque certains événements se produisent dans Unica Plan. Vous pouvez utiliser ces procédures pour exécuter les fonctions Unica Plan à partir de systèmes ou de programmes externes.

L'interface API interagit avec Unica Plan Unica Plan au niveau du programme, de la même façon que vous utilisez l'interface graphique comme interface avec au niveau utilisateur. L'API vous permet de construire des procédures. Ces dernières vous permettent d'établir une communication entre Unica Plan et les systèmes externes. Le service Web Unica Plan est l'objet conteneur de ces procédures, de l'API et des déclencheurs.

L'architecture de Unica Plan Integration Services vous est présentée ici.

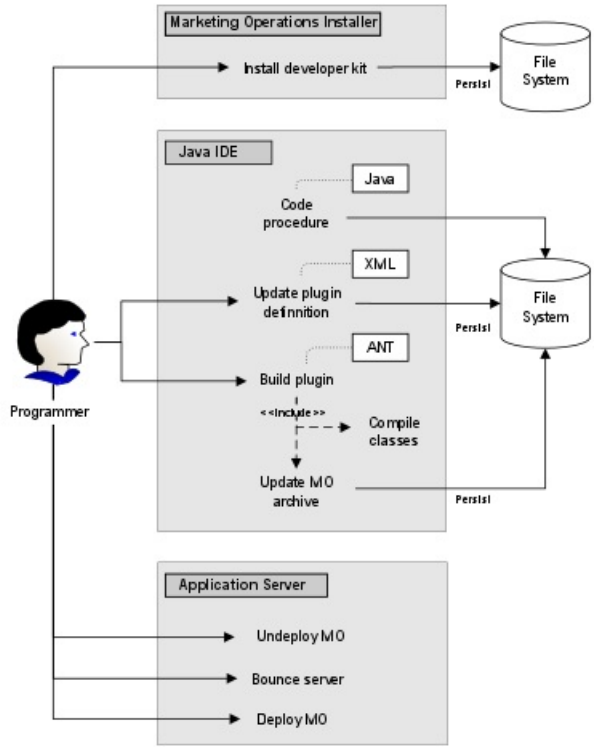


Voici les principaux composants des services Integration Services :

- Unica Plan Procedure Manager : étend la logique métier en interagissant avec Unica Plan via l'API.
- Unica Plan Trigger Manager : associe une condition (par exemple, le changement d'état d'un projet marketing) à une action (une procédure à exécuter lorsque la condition associée au déclencheur est remplie).

Méthodes

Vous utilisez les composants d'Unica Plan Integration Services pour développer des procédures personnalisées, comme l'indique le diagramme suivant :



Après avoir installé le kit de développement, vous effectuez les étapes de base suivantes :

1. Codification de la procédure personnalisée.
2. Mise à jour de la définition du plug-in dans le fichier de définition XML.
3. Génération du plug-in :
    a. Compilation des classes nécessaires.
    b. Si vous utilisez une bibliothèque tiers qui ne fait pas partie de l'archive Unica Plan, intégrez la bibliothèque dans le fichier `plan.war` et procédez au redéploiement.

4. Redémarrez Unica Plan. Les modifications apportées aux classes de procédure sont appliquées lorsque vous redémarrez le serveur d'application.

   Remarque : Si vous modifiez le fichier **plan.war**, vous devez annuler le déploiement et redéployer Unica Plan avec le nouveau fichier **plan.war**. Annulez le déploiement et redéployez Unica Plan si vous utilisez une bibliothèque tiers qui ne fait pas partie de l'archive Unica Plan et que vous modifiez le fichier **plan.war**.

Exemple de base de communication entre Unica Plan et l'API

L'exemple suivant décrit l'établissement d'une communication entre l'API et Unica Plan. Il ne s'agit pas d'un travail utile mais d'un aller-retour entre Unica Plan et Integration Services.

Cet exemple utilise des portions des exemples de procédures fournis avec le kit de développement de Unica Plan Integration Services. Vous pouvez trouver le code référencé ici dans les fichiers suivants :

- `PlanClientFacade.java`
- `PlanWSNOOPTestCase.java`

La méthode `noop` est un appel de service Web en direction de Unica Plan. Elle est définie dans la classe PlanClientFacade et transmet des valeurs nulles dans une matrice.

```
public ProcedureResponse noop(String jobId)
  throws RemoteException, ServiceException {
  NameValueArrays parameters =
    new NameValueArrays(null, null, null, null, null, null, null, null);
  return _serviceBinding.executeProcedure("uapNOOPProcedure", jobId,
 parameters);
}
```

La procédure testExecuteProcedure appelle la méthode `noop` à partir de PlanClientFacade pour établir un aller-retour avec l'application Unica Plan.

```
public void testExecuteProcedure() throws Exception {
  // Time out after a minute
```

```
  int timeout = 60000;
  PlanClientFacade clientFacade = new PlanClientFacade(urlWebService,
 timeout);
  System.out.println("noop w/no parameters");
  long startTime = new Date().getTime();
  ProcedureResponse response = clientFacade.noop("junit-jobid");
  long duration = new Date().getTime() - startTime;

  // zero or positive status => success
  System.out.println("Status: " + response.getStatus());
  System.out.println("Duration: " + duration + " ms");
  assertTrue(response.getStatus() >= 0);
  System.out.println("Done.");
}
```

Pour plus d'informations sur NameValueArrays, ProcedureResponse et les autres méthodes et types de données listés, voir Unica Plan Integration Module et les fichiers JavaDoc.

Installation des services d'intégration

Le module des services d'intégration Unica Plan est un composant distinct et payant. Si vous achetez ce module, vous devez l'installer.

1. Téléchargez les programmes d'installation des services d'intégrationUnica Plan .
2. Les programmes d'installation Unica détectent le module des services d'intégration.
3. Le programme d'installation définit les propriétés de configuration sous **Plan | umoConfiguration | integrationServices | enableIntegrationServices**. Vous pouvez personnaliser l'installation en modifiant les paramètres de configuration. Pour plus d'informations, voir Paramètres de configuration *(à la page 23)*.

Contenu du kit du développeur

Le kit de développement de logiciels comporte une documentation qui contient toutes les classes et interfaces publicapi, ainsi qu'un exemple de code.

Pour l'API SOAP, tous les composants de Unica Plan Integration Services sont installés dans un dossier dont le nom est `devkits`.

Le code exemple est installé dans les dossiers suivants :

- Le dossier **build**, qui contient les scripts permettant de générer et de déployer les procédures personnalisées.
- Le dossier **Classes**, qui contient les classes de procédures compilées.

  Les utilisateurs doivent déployer les classes compilées de leurs procédures personnalisées dans le répertoire spécifié par le paramètre de configuration **integrationProcedureClasspathURL**. Ensuite, Unica Plan Procedure Manager les charge comme spécifié dans le fichier de configuration `procedure-plugins.xml`.
- Le dossier **lib**, qui contient les bibliothèques requises pour le développement et la compilation des procédures personnalisées.
- Le dossier **src**, qui contient les fichiers source pour les procédures personnalisées. Les utilisateurs peuvent stocker à cet emplacement les procédures personnalisées à utiliser en tant que déclencheurs ou services Web. L'API SOAP seulement prend en charge les procédures personnalisées.
  - Le dossier **src/procedure**, qui contient le fichier de configuration `procedure-plugins.xml`. Chaque procédure personnalisée exécutée en tant que déclencheur en fonction d'un événement ou via un service Web externe doit avoir une entrée dans ce fichier. Les entrées doivent contenir un chemin d'accès complet aux classes de procédures et aux paramètres d'initialisation requis.
  - Le dossier **src/procedure**, qui contient également des procédures exemple, incluses dans Unica Plan. Ces procédures peuvent être utilisées pour comprendre et développer vos propres procédures personnalisées.

    Placez les procédures personnalisées dans une nouvelle structure de dossier sous le répertoire **src**, par exemple : `com/<ma_société>/<mon_package>`. Ne placez pas les procédures personnalisées dans le dossier des procédures exemple.
  - Le dossier **src/soap** contient les clients de service Web exemple développés en Java. Utilisez ces exemples comme point de départ pour développer les clients de service Web pour les services d'intégration. Ce dossier contient également

des scripts binaires permettant de démarrer les clients exemple via la ligne de commande.

Fichiers JavaDoc hébergés

Pour obtenir des informations spécifiques sur les méthodes API publiques, reportez-vous à la classe iPlanAPI dans les fichiers de documentation API JavaDoc.

Ces fichiers sont mis à disposition des différentes façons suivantes :

- Via les fichiers du répertoire `<HCL_Unica>/<Plan_Home>/devkits/integration/javadocs` pour l'API SOAP sur le serveur qui héberge Unica Plan.
- Via la procédure suivante : connectez-vous à Unica Plan et sélectionnez **Aide > Documentation sur le produit** à partir de n'importe quelle page, puis téléchargez le fichier `<version>PublicAPI.zip` pour l'API SOAP.

Documentation et aide relatives à Unica Plan

Différentes personnes de votre organisation utilisent Unica Plan pour exécuter différentes tâches. Les informations sur Unica Plan sont disponibles dans un ensemble de guides, chacun d'eux étant destiné à être utilisé par des membres de l'équipe ayant des objectifs et des compétences spécifiques.

Le tableau ci-après présente les informations disponibles dans chaque guide.

Tableau 1. Guides de la documentation Unica Plan

Le tableau à trois colonnes suivant décrit les tâches dans une colonne, les noms des guides dans la deuxième et le public visé dans la troisième.

Si vous	Voir	Public visé
• Planifiez et gérez des projets • Etablissez des tâches, des jalons et du personnel de workflow	Unica PlanGuide d'utilisation	• Chefs de projet • Concepteurs • Directeurs marketing publipostage

Si vous	Voir	Public visé
• Assurez le suivi des dépenses d'un projet • Obtenez des révisions et des approbations de contenu • Générez des rapports • Créer des tâches et des listes de contrôle		• Spécialistes du marketing
• Concevez des modèles, des formulaires, des attributs et des indicateurs • Personnalisation de l'interface utilisateur • Définissez des niveaux d'accès utilisateur et des mesures de sécurité • Implémentez des fonctions facultatives • Configurez et réglez Unica Plan	Unica Plan Guide d’administration	• Chefs de projet • Administrateurs informatiques • Conseillers en matière d'implémentation
• Créez des campagnes marketing • Planifiez des offres • Implémentez l'intégration entre Unica Plan et Unica Campaign • Implémentez l'intégration entre Unica Plan et IBM Digital Recommendations	Unica Plan et Guide d'intégration	• Chefs de projet • Spécialistes d'exécution marketing • Directeurs marketing direct

Si vous	Voir	Public visé
• Découvrez les nouvelles fonctions système • Recherchez les problèmes et contournement connus	Unica PlanEdition Notes®	Toute personne qui utilise Unica Plan
• Installer Unica Plan • Configurer Unica Plan • Procédez à une mise à niveau vers Unica Plan	Unica Plan Guide d'installation	• Conseillers en matière d'implémentation de logiciels • Administrateurs informatiques • Administrateurs de base de données
Créez des procédures personnalisées pour intégrer Unica Plan à d'autres applications	Unica Plan Integration Module et l'API JavaDocs, disponibles lorsque vous cliquez sur **Aide > Documentation sur le produit** dans Unica Plan, puis que vous téléchargez le fichier `UnicaPlan<version>PublicAPI.zip` pour l'API SOAP et `UnicaPlan<version>PublicAPI-RestClient.zip` pour l'API REST.	• Administrateurs informatiques • Administrateurs de base de données • Conseillers en matière d'implémentation
Découvrir la structure de la base de données Unica Plan	Unica Plan Schéma système	Administrateurs de base de données
Avez besoin d'informations supplémentaires pendant que vous travaillez	• Accédez à l'aide et recherchez les manuels suivants : - *Guide d'utilisation, Guide d'administration*, ou - *Guide d'installation* :	Toute personne qui utilise Unica Plan

Si vous	Voir	Public visé
	Cliquez sur **Aide > Aide pour cette page** • Accédez à tous les guides Unica Plan : cliquez sur **Aide > Documentation sur le produit** • Accédez aux guides concernant tous les produits Unica : cliquez sur **Aide > Ensemble de la documentation Unica Suite**	

Chapitre 2. Service Web Unica Plan Integration

Le service Web fournit une vue client de Unica Plan Integration Services, qui fait partie du déploiement du serveur Unica Plan . Le service est utilisé en même temps que les utilisateurs Web de Unica Plan.

Le service Web prend en charge un appel API, executeProcedure.

C'est un client qui effectue directement cet appel de service Web.

Langage WSDL associé à Unica Plan Integration Services

Le langage WSDL a été défini manuellement et constitue le point final de la définition du service Web.

Axis

Cette version du service Web utilise Axis2 1.5.2 pour générer les classes côté serveur qui constituent la mise en œuvre du service web à partir du fichier WSDL. Les utilisateurs peuvent employer n'importe quelle version de Axis ou une technique autre que Axis, pour créer une mise en œuvre côté client permettant une intégration avec l'API à partir du WSDL fourni.

Version du protocole

La version du protocole est explicitement liée au WSDL :

- Dans le nom WSDL, par exemple, `PlanIntegrationService1.0.wsdl`
- En tant que partie du targetNamespace WSDL, par exemple, `xmlns:tns="http://webservices.unica.com /MktOps/services/PlanIntegrationServices1.0?wsdl"`

langage WSDL

Un fichier WSDL est fourni avec Unica Plan Integration Services : `PlanIntegrationServices1.0.wsdl`. Ce fichier WSDL est situé dans le répertoire `integration/examples/soap/plan`. L'exemple de script de génération utilise ce fichier pour générer les modules de remplacement côté client appropriés à connecter au service Web.

executeProcedure

executeProcedure est l'appel d'API pris en charge par le service Web.

Syntaxe

```
executeProcedure(string key, string jobid, NameValueArrays paramArray)
```

Renvoie

```
int: status
Message[]: messages
```

Description

Cette méthode appelle la procédure spécifiée avec une matrice de paramètres facultatifs. L'appel s'exécute de façon synchrone, c'est-à-dire qu'il bloque le client et renvoie le résultat à l'achèvement de l'exécution.

Paramètres

Tableau 2. Paramètres executeProcedure

Nom	Description
key	Clé unique de la procédure à exécuter. Une erreur *RemoteException* est renvoyée si aucune procédure n'est liée à **key**.
ID de travail	Chaîne facultative qui identifie le travail associé à l'exécution de cette procédure. Cette chaîne est un élément passe-système mais elle peut

Nom	Description
	être utilisée pour lier des tâches client à l'exécution d'une procédure particulière.
paramArray	Une matrice de paramètres à transmettre à la procédure. Un état et un message d'erreur sont renvoyés si un ou plusieurs des paramètres sont non valides (par exemple, type non valide ou valeur incorrecte). C'est au client qu'il revient de déterminer les paramètres, leur type et le nombre de valeurs requises par la procédure.

Paramètres de retour

Tableau 3. Paramètres de retour de executeProcedure

Nom	Description
Etat	Code entier : • 0 indique que l'exécution de la procédure a abouti • un entier indique une erreur Les procédures peuvent utiliser l'état pour indiquer différents niveaux d'erreur.
messages	Une matrice de zéro ou plusieurs structures de données de message. Si **status** a pour valeur 0, cette matrice ne contient pas de messages d'ERREUR mais peut contenir des messages d'INFORMATION et d'AVERTISSEMENT. Si **status** est différent de zéro, les messages peuvent contenir un mélange de messages d'ERREUR, d'INFORMATION et d'AVERTISSEMENT.

Type de données de service Web Unica Plan Integration

Les types de données utilisés par le service Web, indépendamment d'une liaison de service ou d'une mise en œuvre de programme particulière.

La notation suivante est utilisée :

- <*type*> : <*type definition*> définit un type de données simple. Par exemple :

  Descripteur : chaîne
- <*type*> : [<*type definition*>] définit un type de données complexe ou une structure de données.
- <*type*> : { <*type definition*> } définit un type de données complexe ou une structure de données.

Les éléments de type complexe et les paramètres API peuvent utiliser ces types pour déclarer des matrices. Par exemple :

```
Handle [] handles
```

Le type, handles, est une matrice de types Handle.

Types primitifs

Les types primitifs sont limités aux types définis dans la table qui suit pour simplifier la prise en charge des liaisons SOAP 1.1. Tous les types peuvent être déclarés sous forme de matrices, par exemple, **String [ ]**. Fondamentalement, les types de données binaires tels que **long** peuvent être représentés sous forme de chaînes par une liaison de protocole (par exemple, SOAP). Cependant, cette représentation n'a aucun effet sur la sémantique du type, les valeurs admises, etc., tels qu'ils sont vus par le client.

Tableau 4. Types primitifs

Type API	Description	Type SOAP	Java™ Type
Booléen	Valeur booléenne : **true** ou **false**	xsd:Boolean	Booléen
dateTime	Valeur de date/heure	xsd:datetime	Date
décimal	Valeur décimale à précision arbitraire	xsd:decimal	java.math.BigDecimal
double	Valeur décimale signée à double précision	xsd:double	double
int	Valeur de type entier signée 32 bits	xsd:int	int

Type API	Description	Type SOAP	Java™ Type
entier	Valeur de type entier signée à précision arbitraire	xsd:integer	java.math.BigInteger
long	Valeur de type entier signée 64 bits	xsd:long	long
chaîne	Chaîne de caractères Unicode	xsd:string	java.lang.String

MessageTypeEnum

```
MessageTypeEnum: { INFORMATION, WARNING, ERROR }
```

MessageTypeEnum est un type énuméré qui définit tous les types de message possibles.

- INFORMATION: message d'information
- ATTENTION : message d'avertissement
- ERREUR : message d'erreur

Message

```
Message: [MessageTypeEnum type, string code, string localizedText, string
 logDetail]
```

Message est une structure de données qui définit le résultat d'un appel API de service Web. Il fournit des zones facultatives pour le code non localisé, le texte localisé et le détail du journal. Actuellement, tous les textes localisés utilisent la langue définie pour l'instance de serveur Unica Plan.

Tableau 5. Paramètres du message

Paramètre	Description
type	MessageTypeEnum définissant le type du message.
code	Code facultatif au format chaîne (string) pour le message.
localizedText	Chaîne de texte facultative à associer au message.
logDetail	Message de trace de pile facultatif.

NameValue

```
NameValue: [string name, int sequence]
```

NameValue est un type complexe de base qui définit une paire nom-valeur. Il définit également une séquence facultative utilisée par le service pour construire les matrices de valeur nécessaires (les séquences sont de base zéro).

Tous les NameValues portant le même nom mais possédant des numéros de séquence différents sont convertis dans une matrice de valeurs et associés au nom commun.

La taille de la matrice dépend du numéro de séquence maximal. Les éléments de matrice non spécifiés possèdent la valeur NULL. Les numéros de séquence de matrice doivent être uniques. La valeur et son type sont fournis par le type étendu.

Tableau 6. Paramètres de NameValue

Paramètre	Description
nom	Chaîne qui définit le nom d'un type NameValue.
séquence	Entier de base zéro qui définit le numéro de séquence de la valeur NameValue concernée.

Les types NameValue étendus sont définis pour chaque type primitif, comme suit :

Tableau 7. Types NameValue étendus

Type étendu	Description
BigDecimalNameValue: NameValue [decimal value]	Type NameValue dont la valeur est un nombre décimal à précision arbitraire.
BigIntegerNameValue: NameValue [integer value]	Type NameValue dont la valeur est un entier signée de façon arbitraire.
BooleanNameValue: NameValue [Boolean value]	Type NameValue dont la valeur est un booléen.
CurrencyNameValue: NameValue [string locale, decimal value]	Type NameValue convenant pour représenter les devises dans une langue spécifique. La langue est représentée par un code de langue ISO, c'est-à-dire un code à deux lettres en minuscules, tel que défini par la norme ISO-639.

Type étendu	Description
	Actuellement, la langue doit correspondre à la langue définie dans l'instance de serveur Unica Plan.
DateNameValue: NameValue [datetime value]	Type NameValue dont la valeur est une date.
DecimalNameValue: NameValue [double value]	Type NameValue dont la valeur est un nombre décimal à double précision.
IntegerNameValue: NameValue [long value]	Type NameValue dont la valeur est un entier de 64 bits.
String NameValue: NameValue [string value]	Type NameValue dont la valeur est une chaîne.

Une matrice des types NameValue étendus est définie afin d'être utilisée lorsque vous avez besoin de définir un jeu de NameValues de différents types.

```
     NameValueArrays: [
BooleanNameValue[]    booleanValues,
StringNameValue[]     stringValues,
IntegerNameValue[]    integerValues,
BigIntegerNameValue[] bigIntegooleanNameValue,
DecimalNameValue[]    decimalValues,
BigDecimalNameValue[] bigDecimalValues
DateNameValue[]       dateNameValues
CurrencyNameValue[]   currencyValues
     ]
```

Chapitre 3. Procédures Unica Plan

Une "procédure" est une classe Java personnalisée ou standard, hébergée par Unica Plan, qui exécute une unité de travail. Les procédures permettent aux clients et aux services Professional Services d'étendre la logique métier de façon arbitraire.

Les procédures suivent un modèle de programmation simple avec une API bien définie pour affecter des composants gérés par Unica Plan. La reconnaissance des procédures s'effectue via un mécanisme de recherche simple et un fichier de définition XML. Unica Plan exécute les procédures en fonction des besoins de ses "clients". (par exemple, en réponse à une demande d'intégration (entrante) ou à l'action d'un déclencheur (interne ou sortant)).

Les procédures s'exécutent de façon synchronisée avec leur client. Les résultats sont directement mis à la disposition du client via un mécanisme d'audit persistant. L'exécution d'une procédure peut également provoquer d'autres événements et déclencheurs dans Unica Plan.

Les procédures doivent être écrites en Java.

Hypothèses

Les classes d'implémentation de procédure sont regroupées dans une arborescence de classes différente ou dans un autre fichier JAR et sont mises à la disposition d'Unica Plan via un chemin URL.

Implémentation de procédure

Le gestionnaire d'exécution de procédure utilise un chargeur de classe indépendant pour charger ces classes en fonction des besoins. Par défaut, Unica Plan recherche dans le répertoire suivant :

`<Plan_Home>/devkits/integration/examples/classes`

Pour changer ce paramètre par défaut, définissez le paramètre **integrationProcedureClasspathURL** sous **Paramètres > Configuration> Plan > umoConfiguration > integrationServices**.

Le nom de la classe de mise en œuvre de procédure obéit aux conventions de dénomination Java acceptées, afin d'éviter des collisions de package avec "unica" et avec les classes des autres fournisseurs. Les clients ne doivent pas placer des procédures dans l'arborescence de packages "com.unica" ou "com.unicacorp".

La mise en œuvre de procédure est codée dans la version Java Runtime utilisée par Unica Plan sur le serveur d'applications (au minimum JRE 1.8).

La classe d'implémentation de procédure est chargée par la règle de chargement de classe qui est normalement utilisée par Unica Plan (généralement **parent-last**). Le serveur d'applications peut fournir des outils et options de développement pour recharger les classes qui pourraient s'appliquer aux procédures Unica Plan, mais cela n'est pas obligatoire.

Bibliothèques

Unica Plan fournit des bibliothèques Open Source et tierces ; les serveurs d'applications utilisent également différentes versions de ces bibliothèques.

En général, cette liste varie d'une version à une autre. Les bibliothèques tierces suivantes sont prises en charge :

- `activation.jar`
- `axiom-api-1.2.15.jar`
- `axiom-compat-1.2.15.jar`
- `axiom-dom-1.2.15.jar`
- `axiom-impl-1.2.15.jar`
- `axis2-adb-1.5.2.jar`
- `axis2-adb-codegen-1.5.2.jar`
- `axis2-codegen-1.5.2.jar`
- `axis2-kernel-1.5.2.jar`
- `axis2-transport-http-1.5.2.jar`
- `axis2-transport-local-1.5.2.jar`
- `httpcore-4.0.jar`
- `commons-codec.jar`
- `commons-httpclient-3.1.jar`

- commons-lang.jar
- commons-logging.jar
- disruptor-3.4.2.jar
- geronimo-stax-api_1.0_spec-1.0.1.jar
- httpclient-4.3.6.jar
- httpcore-4.3.3.jar
- jersey-client-1.17.jar
- jersey-core-1.17.jar
- jersey-json-1.17.jar
- junit-4.4.jar
- log4j.jar
- log4j-api-2.8.2.jar
- log4j-core-2.8.2.jar
- mail.jar
- neethi-2.0.4.jar
- wsdl4j-1.6.2.jar
- xlxpScanner.jar
- xlxpScannerUtils.jar
- xlxpWASParsers.jar
- XmlSchema-1.4.3.jar
- Unica Plan APIs latest version (affinium_plan.jar)
- Unica Platform APIs latest version (unica-common.jar)

Si une procédure ou les classes secondaires importées par la procédure utilisent ces packages, leur utilisation doit être totalement conforme à celle des packages fournis par Unica Plan ou par le serveur d'applications. Dans ce cas, il est nécessaire de retravailler votre code de procédure si une version ultérieure de Unica Plan met à niveau ou abandonne une bibliothèque.

Procédures et unités d'exécution

La procédure doit autoriser les unités d'exécution multiples concernant son propre état. Cela signifie que sa méthode d'exécution ne peut pas dépendre des changements d'état

internes d'un appel à un autre. Une procédure ne peut pas créer des unités d'exécution par elle-même.

Paramètres de configuration

Lorsque vous installez le module d'intégration Unica Plan, trois propriétés de configuration sont définies. Vous pouvez modifier les propriétés de configuration afin de personnaliser le comportement du module d'intégration.

Les propriétés de configuration du module d'intégration sont sous **Plan | umoConfiguration | integrationServices**.

- La propriété de configuration **enableIntegrationServices** permet d'activer ou de désactiver le module du service d'intégration.
- Le paramètre **integrationProcedureDefinitionPath** contient le chemin d'accès complet au fichier XML de définition de procédure personnalisée.

  La valeur par défaut est `<HCL_Unica_Home><Plan_Home>/devkits/integration/examples/src/procedure/procedure-plugins.xml/`.
- Le paramètre **integrationProcedureClasspathURL** contient l'adresse URL du chemin d'accès aux classes pour les procédures personnalisées.

  La valeur par défaut est `file:///<HCL_Unica_Home><Plan_Home>/devkits/integration/examples/classes/`.

  Remarque : Le caractère '/' à la fin du chemin du paramètre `integrationProcedureClasspathURL` est requis pour que le chargement des classes de procédures s'effectue correctement.

Conception

La classe d'implémentation de procédure utilise l'API Unica Plan Unica Plan pour lire et mettre à jour les composants , les services de lancement, etc. D'autres packages Java peuvent être utilisés pour l'exécution d'autres tâches.

Lors de la phase de conception, vous devez vous concentrer sur la production d'une unité de travail unique qui fonctionne de façon atomique. Idéalement, une procédure exécute des séries de tâches qui peuvent être planifiées de façon asynchrone pour être exécutées ultérieurement. Ce modèle d'intégration de type "lancer et oublier" permet d'obtenir une charge minimale sur chacun des deux systèmes.

Remarque : Seuls les classes et les méthodes documentées seront prises en charge dans les futures versions de Unica Plan. Vous devez considérer toutes les autres classes et méthodes de Unica Plan comme non autorisées.

Une fois que vous avez codé et compilé les classes d'implémentation de procédure, vous les mettez à la disposition de Unica Plan. Les scripts de génération qui sont fournis avec la fonction Unica Plan Integration Services placent les procédures compilées à l'emplacement par défaut. L'étape de développement final consiste à mettre à jour le fichier de définition du plug-in de procédure personnalisée qui est utilisé par Unica Plan pour reconnaître les procédures personnalisées.

La procédure doit implémenter l'interface **com.unica.publicapi.plan.plugin.procedure.IProcedure** et comporter un constructeur dans paramètre (modèle JavaBeans habituel). La procédure de codification et de compilation de chaque procédure est effectuée dans un environnement IDE Java choisi par le client (par exemple, Eclipse, Borland JBuilder ou Idea). Un exemple de code est fourni avec Unica Plan sous forme de kits d'outils de développement à l'emplacement suivant :

`<Plan_Home>/devkits/integration/examples/src/procedure`

Cycle de vie de la procédure

Chaque procédure s'exécute via un cycle de vie complet.

Le cycle de vie d'exécution d'une procédure comprend les étapes suivantes :

1. Reconnaissance et initialisation
2. Sélection (facultatif)
3. Exécution
4. Destruction

Reconnaissance et initialisation

Unica Plan doit être informé de toutes les procédures personnalisées et standard disponibles pour une instance d'installation particulière. Ce processes est appelé détection.

Remarque : Les procédures standard (procédures définies par l'équipe d'ingénierie Unica Plan) sont connues implicitement et ne nécessitent donc pas de reconnaissance.

Les procédures personnalisées sont définies dans le fichier de définition du plug-in de procédure. Le gestionnaire du plug-in Unica Plan lit ce fichier lors de l'initialisation. Pour chaque procédure détectée, le gestionnaire de plug-in effectue les tâches suivantes :

1. Instanciation de la procédure ; passage de son état à INSTANCIEE.
2. Création d'un enregistrement d'audit de procédure.
3. Si la procédure a été instanciée, sa méthode **initialize()** est appelée avec tout paramètre d'initialisation trouvé dans son fichier de description de plug-in. Si cette méthode émet une exception, le statut est consigné et la procédure est abandonnée. Dans le cas contraire, la procédure passe à l'état INITIALISEE. Elle est alors prête à être exécutée.
4. Création d'un enregistrement d'audit de procédure.
5. Si la procédure a été initialisée, sa méthode **getKey()** est appelée pour identifier la clé utilisée par les clients pour référencer la procédure. Cette clé est associée à l'instance et sauvegardée pour une recherche ultérieure.

Sélection

De temps en temps, il peut arriver que Unica Plan présente une liste des procédures disponibles aux utilisateurs, par exemple, pour permettre aux administrateurs de définir un déclencheur. Unica Plan présente cette liste uniquement une fois que la procédure a été initialisée, via ses méthodes **getDisplayName()** et **getDescription()**.

Exécution

Une fois la procédure initialisée, Unica Plan reçoit une demande d'exécution de la procédure. Cela peut se produire en même temps que pour d'autres procédures (ou pour la même) s'exécutant sur d'autres unités d'exécution.

Au moment de l'exécution, le gestionnaire d'exécution de procédure effectue les tâches suivantes :

1. Démarrage d'une transaction de base de données.
2. Définition de l'état de la procédure sur EN COURS D'EXECUTION.
3. Création d'un enregistrement d'audit de procédure.
4. Appel de la méthode **execute()** de la procédure avec un contexte d'exécution et tout paramètre d'exécution fourni par le client. La mise en oeuvre de la méthode utilise l'API Unica Plan si nécessaire, en acquérant les verrous d'édition et en transmettant le contexte d'exécution. Si la méthode d'exécution émet une exception, le gestionnaire d'exécution marque la transaction comme devant être annulée.
5. Validation ou annulation de la transaction en fonction des résultats de l'exécution ; définition de l'état de la procédure sur EXECUTEE.
6. Libération de tout verrou d'édition en suspens.
7. Création d'un enregistrement d'audit de procédure.

Remarque : La méthode **execute()** ne doit pas modifier les données d'instance de la procédure.

Destruction

A l'arrêt d'Unica Plan , le gestionnaire du plug-in de procédure passe en revue toutes les procédures chargées. Pour chaque procédure détectée, il effectue les tâches suivantes :

1. Appel de la méthode destroy() de la procédure afin de permettre à cette dernière d'effectuer un nettoyage avant la destruction de l'instance.
2. Passage de l'état de la procédure à FINALISEE (elle ne peut pas être exécutée).
3. Création d'un enregistrement d'audit de procédure.
4. Destruction de l'instance de la procédure.

Principales classes Java

Le kit de développement d'intégration fourni contient une série de fichiers Javadoc pour l'API Unica Plan publique et pour les classes de support.

Les classes Java les plus importantes sont répertoriées ci-après :

- IProcedure (com.unica.publicapi.plan.plugin.procedure.IProcedure) : interface que toutes les procédures doivent implémenter. Les procédures ont un cycle de vie bien défini et accèdent à l'API Unica Plan pour effectuer un travail.
- ITriggerProcedure (com.unica.publicapi.plan.plugin.procedure.ITriggerProcedure) : interface que toutes les procédures de déclencheur doivent implémenter (interface de marqueur).
- IExecutionContext (com.unica.publicapi.plan.plugin.procedure.IExecutionContext) : interface d'objet de contexte opaque qui est transmise à la procédure par le gestionnaire d'exécution. Cet objet comporte des méthodes publiques pour la consignation et la gestion des verrous d'édition. La procédure transmet également cet objet à tous les appels PlanAPI.
- IPlanAPI (com.unica.publicapi.plan.api.IPlanAPI) : interface vers l'API Unica Plan. Le contexte d'exécution fournit une méthode **getPlanAPI()** qui extrait la mise en oeuvre appropriée.

Verrouillage des données

Unica Plan utilise un schéma de verrouillage d'édition pessimiste, c'est-à-dire qu'à un moment donné, un seul utilisateur possède des droits de mise à jour sur le instances du composant. Pour l'utilisateur de l'interface graphique, ce verrouillage est effectué au niveau visuel de l'onglet. Dans certains cas, les données sont réservées pour un sous-ensemble d'une instance, par exemple via un onglet de récapitulatif de projet. Dans d'autres cas, les données sont partagées entre pour plusieurs instances, par exemple via l'onglet workflow. Lorsqu'un utilisateur a acquis un verrou, tous les autres utilisateurs ne possèdent plus qu'un accès en lecture seule aux données concernées.

Afin d'éviter que les modifications apportées par une procédure à une instance de composant ou à un groupe d'instances ne soient écrasées par inadvertance par un autre utilisateur, une procédure doit acquérir les verrous appropriés avant la mise à jour des données du composant. C'est l'objet contexte d'exécution transmis à la méthode **execute()** de la procédure qui est utilisé pour verrouiller les données.

Avant de mettre à jour des données, la procédure doit appeler la méthode **acquireLock()** du contexte pour chaque verrou dont elle a besoins. Par exemple, si une procédure doit mettre à jour un projet et le workflow associé, elle doit acquérir des verrous pour ces deux éléments.

Si un autre utilisateur possède déjà un verrou, la méthode **acquireLock()** émet immédiatement une exception **LockInUseException**. Afin de réduire le nombre de collisions, la procédure doit libérer le verrou dès qu'elle met à jour l'objet.

Le gestionnaire d'exécution libère automatiquement tout verrou en suspens lorsque la méthode d'exécution revient. Dans tous les cas, les verrous ne sont détenus qu'à concurrence de la durée de vie de la transaction de base de données. Les verrous expirent si le délai d'attente de la transaction propre à la base de données a été dépassé.

Remarque : Les verrous d'édition sont différents des transactions de base de données.

Transactions de procédure

Le gestionnaire d'exécution de procédure effectue automatiquement en boucle l'exécution de la procédure avec une transaction de base de données, en la validant ou en l'annulant en fonction du résultat de l'exécution de la procédure.

L'exécution en boucle de la procédure et de la transaction de base de données permet d'être sûr que les mises à jour de la base de données Unica Plan ne sont pas visibles des autres utilisateurs tant qu'elles ne sont pas validées, et rend également les mises à jour automatiques.

L'auteur de la procédure doit cependant acquérir les verrous d'édition nécessaires afin d'être sûr que les autres utilisateurs ne puissent pas copier des modifications dans la base de données avant la fin de l'exécution de la procédure.

Communication de procédure

La méthode **execute()** d'une procédure renvoie un code d'état sous forme d'entier à la table d'audit de procédure Unica Plan. La méthode **execute()** d'une procédure peut également

renvoyer zéro ou plusieurs messages qui sont consignés et conservés dans la table d'audit de procédure.

Le client peut également communiquer les informations d'état d'une autre manière.

Consignation des procédures

Unica Plan possède un fichier journal distinct pour les procédures : `<Plan_Home>\logs\system.log`

Le gestionnaire d'exécution de procédure consigne le cycle de vie de chaque procédure et crée des enregistrements d'audit.

- **logInfo()** : un message d'information est écrit dans le journal des procédures.
- **logWarning()** : un message d'avertissement est écrit dans le journal des procédures.
- **logError()** : un message d'erreur est écrit dans le journal des procédures.
- **logException()** : la trace de pile pour l'exception est vidée dans le journal des procédures.

Fichier de définition du plug-in de procédure

Le fichier de définition du plug-in de procédure fichier définit la classe de mise en oeuvre, les métadonnées et d'autres informations relatives aux procédures personnalisées à héberger dans Unica Plan .

Par défaut, la définition du plug-in de procédure est dans le répertoire suivant :

`<Plan_Home>/devkits/integration/examples/src/procedures/procedure-plugins.xml`

Ce fichier est un document XML qui contient les informations présentées ci-après.

Procedures : liste de zéro ou plusieurs éléments **Procedure**.

Procédure : élément qui définit une procédure. Chaque procédure contient les éléments suivants :

- **key** (facultatif) : chaîne qui définit la clé de recherche de la procédure. Cette clé doit être unique entre toutes les procédures standard et personnalisées qui sont hébergées par une instance Unica Plan donnée. Si elle n'est pas définie, elle prend par défaut la valeur de la version qualifiée complète de l'élément **className**. Les noms commençant par la chaîne "uap" sont réservés à Unica Plan .
- **className** (obligatoire) : nom de package qualifié complet de la classe de procédure. Cette classe doit implémenter la classe IProcedure (com.unica.public.plan.plugin.procedure.IProcedure).
- **initParameters** (facultatif) : liste de plusieurs éléments initParameter ou aucun.

  initParameter(facultatif) : paramètre à transmettre à la méthode initialize() de la procédure. Cet élément inclut le nom du paramètre imbriqué, son type et les éléments de valeur.

  - Nom: chaîne qui définit le nom du paramètre
  - type : nom de classe facultatif de la classe d'encapsuleur Java qui définit le type de la valeur du paramètre. Il doit s'agir de l'un des types suivants :
    - java.lang.String (valeur par défaut)
    - java.lang.Integer
    - java.lang.Double
    - java.lang.Calendar
    - java.lang.Boolean
  - Valeur : forme de la chaîne associée à la valeur d'attribut en fonction de son type

Chapitre 4. API SOAP d'Unica Plan

L'API SOAP d'Unica Plan Unica Plan est une façade qui offre une vue client d'une instance de en cours d'exécution.

Nous n'exposerons aux utilisateurs qu'une petite partie des possibilités de Unica Plan. L'API est utilisée simultanément par des utilisateurs Web de Unica Plan et par les demandes et les déclencheurs Unica Plan Integration Services WebService SOAP. Cette API prend en charge les types suivants d'opérations :

- Création et suppression de composant
- Reconnaissance (par type de composant, valeur d'attribut, etc.)
- Inspection de composant (via ses attributs, de liens spécialisés, etc.)
- Modification de composant

Remarque : seuls les administrateurs peuvent utiliser les API de Unica Plan.

Contenu de l'API SOAP d'Unica Plan

Le package com.unica.publicapi.plan.api fournit l'API SOAP d'Unica Plan .

Ce package offre des interfaces et des exceptions et contient les types de classe suivants :

- Des types de données énumérées.
- Des descripteurs qui identifient les instances d'objet et de composant.
- Une mappe Java, AttributeMap.

La documentation complète relative à l'API, y compris toutes les méthodes et toutes les valeurs possibles, est disponible en cliquant sur **Aide > Documentation sur le produit** dans une instance de Unica Plan, puis en téléchargeant le fichier HCL<version>PublicAPI.zip

Interfaces de l'API SOAP

L'interface de programme d'application SOAP d'Unica Plan inclut **IPlanAPI** et **IExecutionContext**.

L'API SOAP de Unica Plan inclut les interfaces suivantes :

IPlanAPI

Définit l'API publique pour Unica Plan. Fournit des méthodes permettant de créer, découvrir et modifier des objets, notamment des dossiers, des projets, des programmes, des tâches de workflow et des membres d'une équipe.

Pour les systèmes sur lesquels l'intégration facultative à Unica Campaign est activée, fournit également des méthodes permettant de créer, découvrir et modifier des offres.

IExecutionContext

Définit les déclencheurs et verrouille ces méthodes d'exécution dans l'API.

Méthodes API

Pour obtenir des informations spécifiques sur les méthodes API publiques, reportez-vous à la classe iPlanAPI dans les fichiers de documentation API JavaDoc.

Pour accéder à ces fichiers, connectez-vous à Unica Plan et sélectionnez **Aide > Documentation produit** dans n'importe quelle page, puis téléchargez le fichier `<version>PublicAPI.zip`.

Exceptions courantes de l'API SOAP

Les exceptions courantes qui sont émises par l'API SOAP sont notamment NotFoundException, AuthorizationException, DataException, InvalidExecutionContextException et NotLockedException.

La liste ci-après explique pourquoi ces exceptions peuvent se produire.

- <*object type*>NotFoundException : le système n'est pas en mesure de renvoyer l'élément ou l'objet spécifié.
- AuthorizationException : l'utilisateur qui est associé au contexte d'exécution n'est pas autorisé à effectuer l'opération demandée. Cette exception peut être émise par n'importe quelle méthode API, par conséquent, elle n'est pas déclarée.

- DataException une exception s'est produite dans la couche de base de données sous-jacente dans Unica Plan. Pour plus d'informations, voir le journal SQL.
- InvalidExecutionContextException: Un problème lié au contexte d'exécution transmis à une méthode API s'est produit (par exemple, la méthode n'a pas été correctement initialisée). Cette exception peut être émise par n'importe quelle API, par conséquent, elle n'est pas déclarée.
- NotLockedException: tentative de mise à jour des données de composant sans acquisition préalable du verrou requis. Voir la méthode acquireLock() de l'interface IExecutionContext.

Descripteurs de l'API SOAP

Un descripteur est un objet URL spécial qui référence une instance d'objet particulière dans une instance d'Unica Plan . Le type de composant, l'identificateur des données interne et une URL de base d'instance sont des descripteurs.

Les descripteurs utilisés ou générés par les API peuvent être externalisés vers une adresse URL complète. Vous pouvez utiliser cette URL de différentes manières, par exemple pour ouvrir une vue du composant dans l'interface graphique Unica Plan, l'envoyer dans des courriers électroniques ou l'utiliser dans une autre procédure en tant que paramètre.

Les descripteurs sont uniquement valides pour une instance de service ou une instance en cluster Unica Plan donnée, mais leur validité est garantie pour toute la durée de vie du service déployé. Par conséquent, les descripteurs peuvent être sauvegardés dans un fichier pour une référence ultérieure, mais ils ne peuvent pas être utilisés pour accéder à des composants sur une autre instance Unica Plan. Cette restriction s'applique également aux instances présentes sur le même serveur hôte physique. Toutefois, Unica Plan ne fournit pas de mécanisme de mappage de différentes URL de base à l'instance en cours en vue de permettre le déplacement d'une instance sur une autre serveur (par exemple, si le matériel présente un dysfonctionnement).

Les descripteurs sont indépendants du client. Par exemple, un déclencheur peut transmettre un descripteur à une procédure, qui l'utilise ensuite en tant que paramètre dans un appel SOAP vers un système tiers. Le système tiers peut ensuite renvoyer une demande SOAP à Unica Plan pour démarrer une procédure de mise à jour d'un attribut.

Les membres de la classe Handle comportent des méthodes de fabrique destinées à créer des descripteurs pour divers types d'URL. Exemples :

Approbation

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=approvaldetail&
approvalid=101
```

Document

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=asset&
assetMode=VIEW_ASSET&assetid=101
```

Dossier de documents

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=folder&id=101
```

Bibliothèque de documents

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=library&id=101
```

Pièce jointe

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=attachmentview&
attachid=101&parentObjectId=101&parentObjectType=project
```

Compte financier

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=accountdetails&
accountid=101
```

Dossier

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=grouping_folder&
folderid=1234
```

Facture

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=invoicedetails&
invoiceid=134
```

Ligne de facture

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=invoicedetails&
invoiceid=134&line_item_id=101
```

Objet marketing

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=componenttabs&
componentid=creatives&componentinstid=1234
```

Grille d'objet marketing

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=componenttabs&
componentid=creatives&componentinstid=1234&gridid=grid
```

Ligne de grille d'objet marketing

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=componenttabs&
componentid=creatives&componentinstid=1234&gridid=grid&gridrowid=101
```

Equipe de plan

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=teamdetails&
func=edit&teamid=100001
```

Utilisateur du plan

```
http://mymachine:7001/plan/affiniumplan.jsp?
cat=adminuserpermissions&
func=edit&userId=101
```

Programme

```
http://mymachine:7001/plan/affiniumplan.jsp?
cat=programtabs&programid=125
```

Grille de programme

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=programtabs&
```

```
programid=1234&gridid=grid
```

Ligne de grille de programme

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=programtabs&
programid=1234&gridid=grid&gridrowid=101
```

Projet

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=projecttabs&
projectid=1234
```

Grille de projet

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=projecttabs&
projectid=1234&gridid=grid
```

Ligne de grille de projet

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=projecttabs&
projectid=1234&gridid=grid&gridrowid=101
```

Ligne de projet

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=projecttabs&
projectid=1234&projectlineitemid=123&projectlineitemisversionfinal=false
```

Etat de workflow

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=projectworkflow&
projectid=1234&taskid=5678
```

Tâche de workflow

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=projectworkflow&
projectid=1234&taskid=5678
```

Classe AttributeMap de l'API SOAP

La classe AttributeMap est une mappe Java qui contient uniquement des attributs. L'attribut <Name> est la clé d'entrée de mappe et la matrice des <values> d'attribut (notez l'emploi du pluriel) est la valeur d'entrée de mappe.

La classe AttributeMap contient les zone suivantes :

- *<Name>* : nom défini par programme de l'attribut. Ce nom sert de clé unique pour accéder à l'attribut dans l'instance de composant où il apparaît.

  Remarque : *<Name>* n'est pas obligatoirement le nom d'affichage présenté à l'utilisateur dans l'interface graphique. Pour les composants créés à partir de modèles (tels que les projets ou les tâches de workflow), le nom d'attribut est spécifié par la définition d'élément du modèle. Ce nom doit être unique. Pour les autres composants, le nom d'attribut est généralement dérivé par voie de programme de l'instance de composant côté serveur (par exemple, via l'introspection Java).

  Remarque : Par convention, les attributs personnalisés incluent le nom du formulaire dans lequel la version modifiable est définie : *<form_name>.<attribute_name>*.
- *Values* : matrice d'objet Java contenant zéro, une ou plusieurs valeurs d'attribut. Le type de chaque valeur doit être identique et en accord avec le type de l'attribut défini dans Unica Plan. Seul l'encapsuleur Java et les types Unica Plan suivants sont pris en charge :
  - AssetLibraryStateEnum : valeur AssetLibraryStateEnum de type énuméré.
  - AssetStateEnum : valeur AssetStateEnum de type énuméré.
  - AttachmentTypeEnum : valeur AttachmentTypeEnum de type énuméré.
  - AttributeMap : mappe qui contient des attributs.
  - BudgetPeriodEnum : valeur BudgetPeriodEnum de type énuméré.
  - BudgetTypeEnum : valeur BudgetTypeEnum enumerated de type énuméré.
  - Descripteur : référence à une instance de composant, une ligne de grille, un attribut, etc.
  - InvoiceStateEnum : valeur InvoiceStateEnum de type énuméré.
  - java.io.File : représentation d'un fichier.

- java.lang.Boolean : valeur booléenne (True ou False)
- java.lang.Double : valeur de nombre décimal à double précision.
- java.lang.Float : valeur de nombre décimal à simple précision.
- java.lang.Integer : valeur de type entier 32 bits
- java.lang.Long : valeur de type entier 64 bits
- java.lang.Object : Object Java générique
- java.lang.String : chaîne comprenant zéro ou plusieurs caractères Unicode
- java.math.BigDecimal : valeur de nombre décimal signée à précision arbitraire. Convient pour les devises ; l'interprétation de la valeur dépend de la langue utilisée pour les devises pour le client.
- java.math.BigInteger : valeur de type entier à précision arbitraire.
- java.net.URL : objet URL.
- import java.util.ArrayList : liste des objets.
- java.util.Calendar : valeur date-heure pour une langue particulière.
- java.util.Date : valeur date-heure. Ce type est obsolète. Utilisez à la place java.util.Calendar ou java.util.GregorianCalendar.

  Remarque : Pour mettre en œuvre la date, les utilisateurs peuvent utiliser java.util.Calendar ou java.util.GregorianCalendar.
- java.util.GregorianCalendar : GregorianCalendar est une sous-classe concrète de java.util.Calendar et fournit un système de calendrier standard utilisé dans la plupart des pays du monde entier.
- MonthEnum : valeurMonthEnum de type énuméré.
- ProjectStateEnum : valeur ProjectStateEnum de type énuméré
- QuarterEnum : valeur QuarterEnum de type énuméré.
- TaskStateEnum : valeur TaskStateEnum de type énuméré.
- WeekEnum : valeur WeekEnum de type énuméré.

Les métadonnées d'un attribut (telles que le nom d'affichage traduit et la description associée) sont définies par le modèle qui est associé à l'attribut et à son instance d'objet parent. Les attributs fournissent un mécanisme simple et extensible pour afficher les attributs d'instance d'objet facultatifs et obligatoires, tels que le nom du projet, le code et la date de début.

Types de données énumérées de l'API SOAP

L'API SOAP d'Unica Plan prend en charge les types de données énumérées et les valeurs qui suivent.

ApprovalMethodEnum

ApprovalMethodEnum définit des méthodes d'approbation valides. Les valeurs possibles sont les suivantes :

- SEQUENTIAL
- SIMULTANE

ApprovalStateEnum

ApprovalStateEnum définit des états d'approbation valides. Les valeurs possibles sont les suivantes :

- CANCELLED
- TERMINEES
- EN COURS
- PAS D'ETAT
- EN ATTENTE

AssetLibraryStateEnum

AssetLibraryStateEnum définit des états de bibliothèque de documents valides. Les valeurs possibles sont les suivantes :

- DESACTIVEE
- ACTIVEE

AssetStateEnum

AssetStateEnum définit des états de document valides. Les valeurs possibles sont les suivantes :

- ARCHIVE

- BROUILLON
- FINALISE
- VERROUILLE

AttachmentTypeEnum

AttachmentTypeEnum définit des types de pièce jointe valides. Les valeurs possibles sont les suivantes :

- RESSOURCE
- FILE
- URL

BudgetPeriodEnum

BudgetPeriodEnum définit les périodes de budget possibles. Les valeurs possibles sont les suivantes :

- ALL
- MENSUEL
- TRIMESTRIEL
- HEBDOMADAIRE
- ANNUEL

BudgetTypeEnum

BudgetTypeEnum définit les types de budget possibles. Les valeurs possibles sont les suivantes :

- REEL
- ALLOUE
- ENGAGE
- PREVU
- TOTAL

ComponentTypeEnum

ComponentTypeEnum identifie les types de composant Unica Plan accessibles. Les valeurs possibles sont les suivantes :

- APPROBATION
- RESSOURCE
- DOSSIER_DOCUMENTS
- BIBLIOTHEQUE_DOCUMENTS
- PIECE JOINTE
- COMPTE_FINANCIER
- DOSSIER_GROUPEMENT
- FACTURE
- OBJET_MARKETING
- EQUIPE_PLAN
- UTILISATEUR_PLAN
- PROGRAM
- PROJECT
- DEMANDE_PROJET
- TACHE
-

InvoiceStateEnum

InvoiceStateEnum définit des états de facture valides. Les valeurs possibles sont les suivantes :

- CANCELLED
- BROUILLON
- PAYE
- PAYABLE

MonthEnum

MonthEnum définit des valeurs de mois valides.

OfferStateEnum

OfferStateEnum définit des états d'offre valides. Les valeurs possibles sont les suivantes :

- OFFRE_ETAT_BROUILLON
- OFFRE_ETAT_PUBLIEE
- OFFRE_ETAT_RETIREE

ProjectCopyTypeEnum

ProjectCopyTypeEnum définit des méthodes valides de copie d'un projet. Les valeurs possibles sont les suivantes :

- COPIE_AVEC_INDICATEURS_PROJET
- COPIE_AVEC_INDICATEURS_MODELE

ProjectParticipantLevelEnum

ProjectParticipantLevelEnum identifie les rôles qui peuvent être attribués aux utilisateurs dans un projet. Les valeurs possibles sont les suivantes :

- OWNER
- PARTICIPANT
- DEMANDEUR

ProjectStateEnum

ProjectStateEnum définit des état de projet et de demande valides. Les valeurs possibles sont les suivantes :

- ACCEPTE
- CANCELLED
- TERMINEES
- BROUILLON
- EN COURS
- EN RAPPROCHEMENT
- LATE : le projet n'a pas démarré à la date de début prévue.

- NON DEMARRE
- EN ATTENTE
- DEPASSE : le projet ne s'est pas terminé avant la date de fin prévue.
- RENVOYE
- SUBMITTED

Pour plus d'informations sur des statuts des projets et des tâches, voir Unica Plan - Guide d'utilisation.

QuarterEnum

QuarterEnum définit des valeurs de trimestre valides : Q1, Q2, Q3 et Q4.

TaskStateEnum

TaskStateEnum définit des états de tâche de workflow valides. Les valeurs possibles sont les suivantes :

- ACTIF
- DESACTIVE
- FINISHED
- EN ATTENTE
- IGNORE

WeekEnum

WeekEnum définit des valeurs de semaine valides sur une année, comprises entre SEMAINE_1 et SEMAINE_53.