



zenika
<animés par la passion>

Gestion du routage

Dispositif de Gestion des Echanges

Version 1.18-SNAPSHOT du 2018-10-21

Sommaire

1. Introduction	3
2. Contexte	3
2.1. Architecture fonctionnelle	3
2.2. Services webMethods	3
2.2.1. Routage Diffusion Messages	3
2.3. Présentation de l'écran de simulation	4
3. Packages webMethods	7
3.1. SgeBpmDf : routage Diffusion Fichier	7
3.1.1. Services publics	7
3.2. SgeBpmDm : routage Diffusion Messages	8
3.2.1. Services publics	9
3.3. Gestion des anomalies	10
3.4. Documents WM	11
3.4.1. sgeCommonsDocs.docs.dm.bpm:Data (non publiable)	11
3.5. SgeBpmCommons	12
3.5.1. Services publics	12
4. Jars	13
4.1. CacheRFO.initRisqueListFromRFO	13
4.2. CacheRFO.getRisqueListFromRFO	13
4.3. GenericRoutingManager.routeWithRestriction	13
4.4. GenericRoutingManager.routeWithRisques	13
4.5. GenericRoutingManager.routeWithComplement	13
4.6. GenericRoutingManager.routeWithAbonnement	14
4.7. GenericRoutingManager.routeWithDCR	14
4.8. GenericRoutingManager.getFinalResult	14
4.9. DfRoutingManager.storeRoutingResult	14
4.9.1. Entrées/Sorties	14
4.9.2. Algorithme	14
4.10. DfRoutingManager.buildEndBpmDfData	15
4.10.1. Entrées/Sorties	15
4.10.2. Algorithme	15
4.11. DmRoutingManager.startMsg	16
4.11.1. Déclenchement	16
4.11.2. Entrées/Sorties	16
4.11.3. Algorithme	17
4.12. DmRoutingManager.startRouteMsg	17
4.12.1. Déclenchement	17
4.12.2. Entrées/Sorties	17

4.12.3. Algorithme	18
4.13. DmRoutingManager.endRouteMsg	19
4.13.1. Déclenchement	19
4.13.2. Entrées/Sorties	19
4.13.3. Algorithme	19
4.14. DmRoutingManager.endMsg	20
4.14.1. Déclenchement	20
4.14.2. Entrées/Sorties	20
4.14.3. Algorithme	20
4.15. DmRoutingManager.storeRoutingResult	20
4.15.1. Entrées/Sorties	21
4.15.2. Algorithme	21
4.16. DmRoutingManager.buildEndBpmDmData	21
4.16.1. Entrées/Sorties	22
4.16.2. Algorithme	22
5. Annexes	24
5.1. Marques de révision	24

Table 1. Historique

Date	Auteur	Détail
2018-08-24	bcouetil	Reworked asciidoc index.html
2018-08-24	bcouetil	Icones added for download + favicon added for webpage
2018-08-23	bcouetil	Initial commit

Table 2. Références

Réf.	Ver.	Date	Document	Description
[01]	01.6	27/04/2017	SGE_V2_SFG_Spécifications_Fonctionnelles_Générales.docx	Spécifications fonctionnelles générales du SGE V2
[02]		2018-10-21	SAF-architecture.pdf	Cadre d'architecture SGE V2, contenant notamment les noms des IS
[03]	01.4	27/04/2017	SGE_V2_DSF_Dossier_de_Spécifications_Fonctionnelles.docx	Spécifications fonctionnelles détaillées du SGE V2
[04]	01.0	29/09/2016	SGE_V2_FS_Diffuser_un_fichier_(DF).docx	Fiche Solution décrivant le routage pour la nature Diffusion Fichier pour le SGE V2
[05]		2018-10-21	STS-Cache-Catalogue.pdf	Description du cache du catalogue SGE V2
[06]	xxx	xxx	Ajouter FS DM Batch	
[07]	xxx	xxx	Ajouter FS DM TR	
[08]	xxx	xxx	SGE_V2_DSF_Liste_des_codes_d_erreurs.xlsx	Liste des codes d'erreurs SGE V2

1. Introduction

Ce document décrit les opérations de routage réalisées par l'IS « MET » de l'application SGE V2 pour les natures d'échanges suivantes :

- Diffusion Fichier
- Diffusion Messages

2. Contexte

2.1. Architecture fonctionnelle

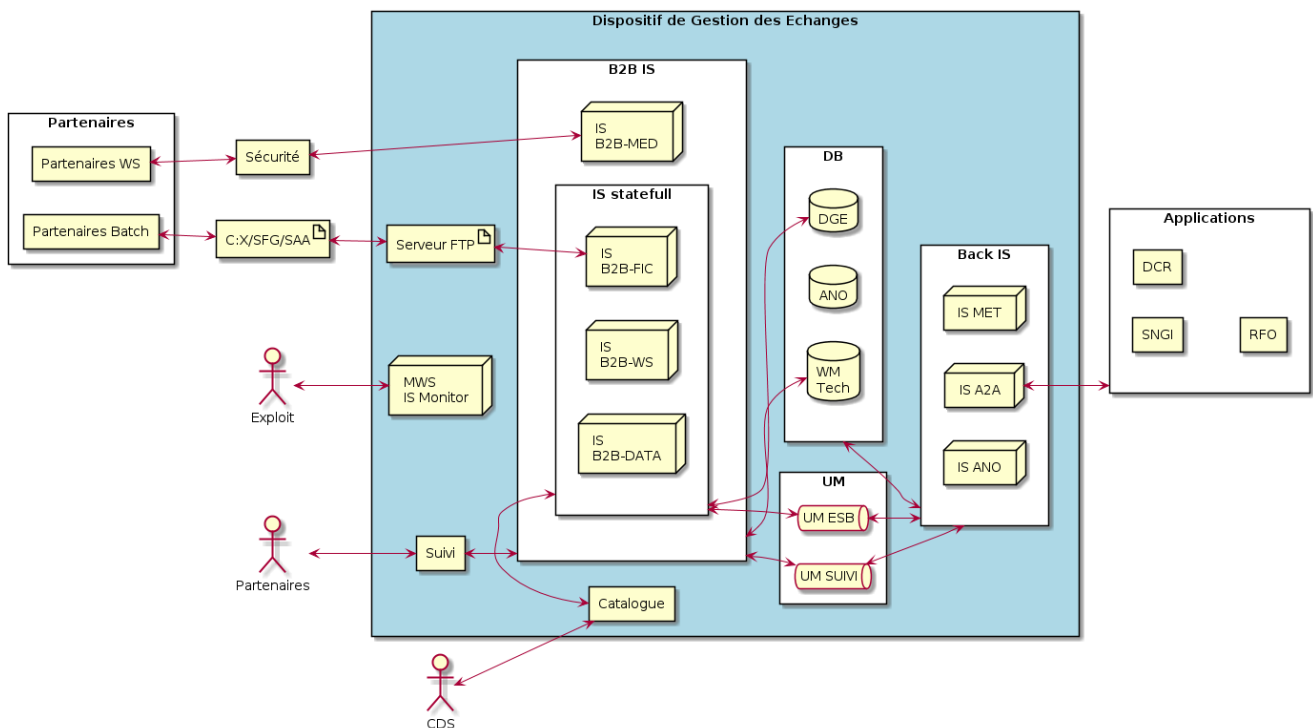


Figure 1. Architecture fonctionnelle

2.2. Services webMethods

2.2.1. Routage Diffusion Messages

Nature : DM▼	Mode : Batch▼
Template : Q8001.xml▼	Charger Supprimer
<pre><?xml version="1.0" encoding="UTF-8"?> <DM_Entree:DGE xmlns:DM_Entree="http://www.cnnav.fr/rncps/DGE/interfaces/DM/v2/DM_Entree"> <Entete> <Nature> <Type>DiffusionMessages</Type> <Version>02.01</Version> <Phase>DM_ENTREE</Phase> </Nature> <Service> <Reference>M0012E001</Reference> <Version>02.01</Version> </Service> <Echange> <Identifiant>21841021AQ178000</Identifiant> <Horodatage>2015-04-14T11:39:15.234</Horodatage> <RefSISO>SISO000000001</RefSISO> <NbMessages>3</NbMessages></pre>	
<input checked="" type="checkbox"/> Remplacer les identifiants	Code testeur : ASO Soumettre
Nouveau template : Q8001 copy.xml	Sauvegarder

Figure 3. Vue conceptuelle du simulateur

- Liste déroulante « **Nature** » permettant de choisir entre « DF », « DM », « ABT CRUD » et « ABT CHGT »
- Liste déroulante « **Mode** », filtrante, avec choix possibles :
 - DF : « Fichier »
 - DM, ABT CRUD, ABT CHGT : « Batch » / « Temps Réel »
- Liste déroulante « **Template** » :
 - Choix du fichier parmi les templates du simulateur côté serveur, filtré par Nature et Canal.
- Bouton [**Charger**] pour charger le fichier choisi.
- Bouton [**Supprimer**] pour supprimer définitivement le fichier choisi des templates, avec confirmation. Ceci permet d'être autonome dans la gestion du patrimoine de test.
- Zone d'aperçu éditable permettant de visualiser le futur fichier injecté, sans écrasement du template.
- Case à cocher « **Remplacer les identifiants et dates** » qui permet de remplacer les identifiants par des valeurs uniques, quelque soit le contenu préalable dans l'aperçu.
- Champ « **Code testeur** » utilisé dans les identifiants si remplacement activé.
- Bouton [**Soumettre**] qui utilise le contenu de l'aperçu comme source et applique les modifications nécessaires, conformément à la configuration du simulateur suivant qu'on ait forcé le remplacement des identifiants avant de l'injecter.
 - Une fois le fichier généré, le nom du fichier apparaît sur l'écran.
- Champ « **Nouveau template** » qui permet d'indiquer sous quel nom le template de contenu de l'aperçu devra être enregistré.
- Bouton [**Sauvegarder**] qui permet de sauvegarder le template.

- Un message indique si la sauvegarde a fonctionné, en proposant l'écrasement si un fichier du même nom existe.

3. Packages webMethods

3.1. SgeBpmDf : routage Diffusion Fichier

Le routage de la nature Diffusion Fichier est réalisé par le package webMethods **SgeBpmDf**.

3.1.1. Services publics

sgeBpmDf.pub:routeFile

Déclenchement

Le module de routage de la Diffusion Fichier se déclenche sur réception du document **sgeCommonsDocs.docs.df.bpm:StartBpmDf** publié par l'IS « B2B-FIC-PE-n », lorsque le résultat des contrôles d'entrée effectués sur le fichier est OK.

Ce document contient les données d'entête du fichier à router :



Sur réception de ce document, le trigger **sgeBpmDf.triggers:routeFile** est déclenché et le service associé démarre.

Traitement

Le service **sgeBpmDf.pub:routeFile** réalise le routage du fichier(cf. algorithmes fonctionnels : doc [04]) :

- Prise en compte de la liste restrictive (*) :
 - appel à [routeWithRestriction](#)
- Prise en compte de la liste des risques :
 - appel à [routeWithRisques](#)
- Prise en compte de la liste complémentaire (*) :
 - appel à [routeWithComplement](#)
- Détermination du résultat final de routage :
 - appel à [getFinalResult](#)
- Stockage des demandes de routage, pour chaque destinataire (*) :
 - appel à [storeRoutingResult](#)
- Construction des données d'ARLE :

- appel à `buildEndBpmDfData`

(*) accès au cache du catalogue, cf. document [05]

En fin de traitement, le service publie le document `sgeCommonsDocs.docs.df.bpm:EndBpmDf`, contenant les données nécessaires à la construction de l'ARLE et des entêtes de SORTIE :



Cycle de vie du traitement

Dans la base de données ESB, l'état est mis à jour au fur et à mesure du traitement dans la table ESB.DEMANDES :

Phases	Etats
ROUTAGE	<ul style="list-style-type: none"> • ENCOURS : traitement en cours • ERREUR : traitement en erreur • OK : traitement terminé, au moins un destinataire trouvé • KO : traitement terminé, aucun destinataire trouvé <p>Dans le cas OK, une ligne pour chaque destinataire est créée dans la table DEMANDE_ROUTAGES : phase INITIALISATION, état OK. Dans le cas KO, une seule ligne est créée (sans destinataire) : phase INITIALISATION, état KO.</p> <pre> graph TD ENCOURS([ENCOURS]) --> ERREUR([ERREUR]) ENCOURS --> KO([KO]) ENCOURS --> OK([OK]) </pre>

3.2. SgeBpmDm : routage Diffusion Messages

Le routage de la nature Diffusion Messages est réalisé par le package webMethods **SgeBpmDm**.

3.2.1. Services publics

sgcCommonsDocs.docs.dm.bpm:StartBpmDm

Déclenchement

Le module de routage de la Diffusion Messages se déclenche sur réception du document **sgcCommonsDocs.docs.dm.bpm:StartBpmDm** publié par l'IS B2B-FIC lorsque le résultat des contrôles d'entrée effectués sur la demande est OK. Ce document contient les informations du message à router.

Cf. paragraphe "Documents WM" pour la description des documents webMethods.

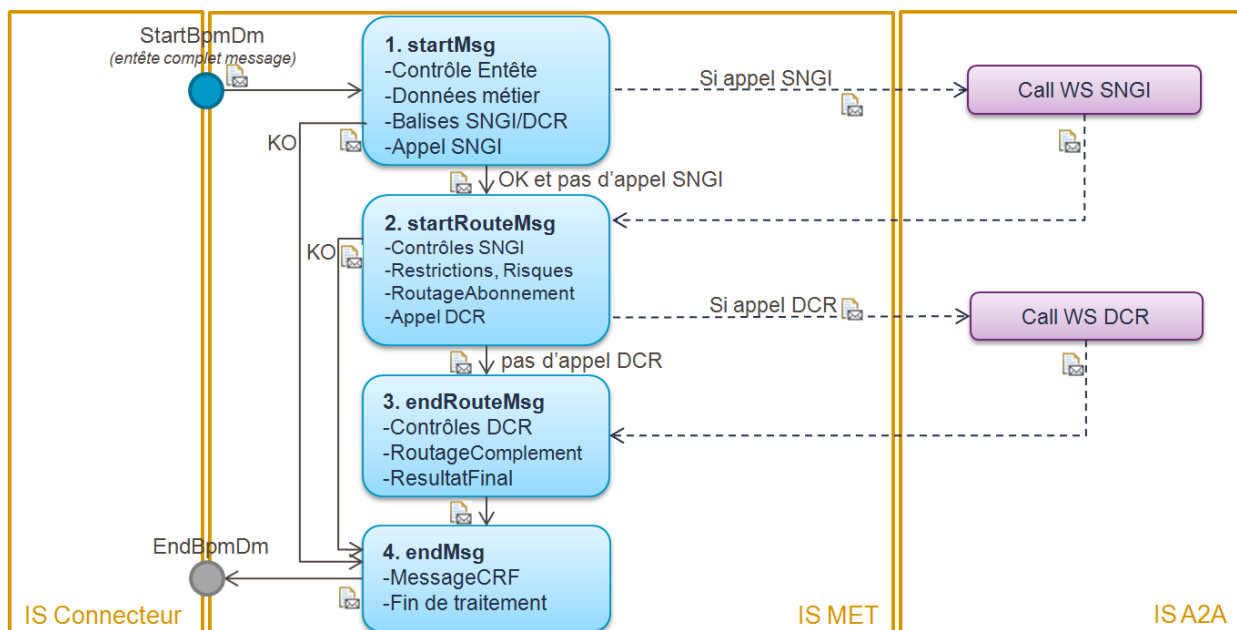
Sur réception de ce document, le trigger **sgcBpmDm.triggers:startMsg** est déclenché et le service associé démarre.

Traitement

Le traitement est réalisé en quatre étapes correspondant aux services suivants :

- **sgcBpmDm.pub:startMsg**: appel à [DmRoutingManager.startMsg](#)
- **sgcBpmDm.pub:startRouteMsg**: appel à [DmRoutingManager.startRouteMsg](#)
- **sgcBpmDm.pub:endRouteMsg**: appel à [DmRoutingManager.endRouteMsg](#)
- **sgcBpmDm.pub:endMsg**: appel à [DmRoutingManager.endMsg](#)

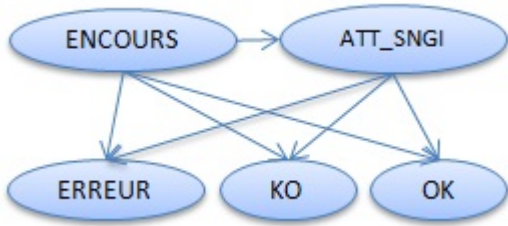
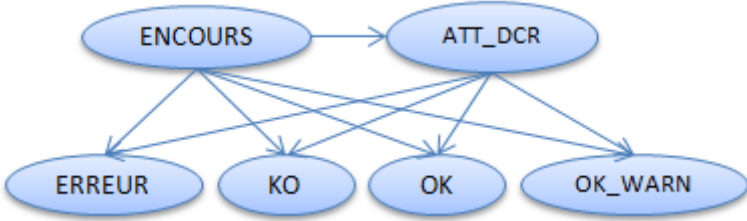
Le schéma suivant représente les enchainements possibles de ces étapes :



En fin de traitement, le service publie le document **sgcCommonsDocs.docs.dm.bpm:EndBpmDm**, signalant la fin de traitement d'un message. Ce document contient les informations nécessaires au connecteur DM pour créer le message de CRF ainsi que les messages de sortie à destination de chaque consommateur.

Cycle de vie du traitement

Dans la base de données ESB, l'état est mis à jour au fur et à mesure du traitement dans la table ESB.MESSAGES :

Phases	Etats
VALIDATION	<ul style="list-style-type: none"> • ENCOURS : traitement en cours • ATT_SNGI : traitement en attente de réponse SNGI • ERREUR : traitement en erreur • OK : traitement terminé, tous les contrôles sont OK • KO : traitement terminé, au moins un contrôle est KO  <pre> graph TD ENCOURS --> ATT_SNGI ENCOURS --> ERREUR ENCOURS --> KO ENCOURS --> OK ATT_SNGI --> ERREUR ATT_SNGI --> KO ATT_SNGI --> OK </pre>
ROUTAGE	<ul style="list-style-type: none"> • ENCOURS : traitement en cours • ATT_DCR : traitement en attente de réponse DCR • ERREUR : traitement en erreur • OK : traitement terminé, aucune erreur de routage relevée • OK_WARN : traitement terminé, au moins une erreur de routage relevée • KO : traitement terminé, aucun destinataire trouvé  <pre> graph TD ENCOURS --> ATT_DCR ENCOURS --> ERREUR ENCOURS --> KO ENCOURS --> OK ENCOURS --> OK_WARN ATT_DCR --> ERREUR ATT_DCR --> KO ATT_DCR --> OK ATT_DCR --> OK_WARN </pre> <p>Dans les cas OK et OK_WARN, une ligne pour chaque destinataire est créée dans la table MESSAGE_ROUTAGES : phase INITIALISATION, état OK. Dans le cas KO, une seule ligne est créée (sans destinataire) : phase INITIALISATION, état KO.</p>

3.3. Gestion des anomalies

En cas d'erreur technique lors du traitement du routage :

- en BDD ESB, l'état **ERREUR** est positionné sur la phase en cours :
 - pour DF : tables DEMANDES et/ou DEMANDE_ROUTAGES
 - pour DM : tables MESSAGES et/ou MESSAGE_ROUTAGES

- un log d'avancement **logERREUR** est émis vers le suivi pour signaler l'erreur.
- une anomalie technique est générée et stockée dans le gestionnaire des anomalies. Si l'erreur est temporaire (exemple : problème d'accès base de données), l'anomalie peut être rejouée et le traitement reprend alors. Si l'erreur est définitive (exemple : donnée incorrecte), l'anomalie ne peut pas être rejouée.

3.4. Documents WM

3.4.1. sgeCommonsDocs.docs.dm.bpm:Data (non publiable)

Ce document contient la structure commune des données partagées par tous les documents webMethods du BPM DM :

- Message
 - Identifiant
 - Horodatage
 - Type
 - CleSelectVersion
 - RefConversation
 - Individu
 - Fonction
- Service : Reference / Version
- Routage
 - Risques : Identifiant[]
 - Restriction[] : Identifiant / Type
 - Complement[] : Identifiant / Type
- Filtrage[] : Key / Value
- Rattachement[] : Identifiant / Etat / DateCloture
- IdentiteIndividu ⇒ Bloc IdentiteIndividu complet (cf. Contrat d'Interface DM)
- Producteur
 - Identifiant/ Type / Libelle
 - Resultat
 - Statut
 - Complement[] : Code / Libelle
 - Option[] : Key / Value
- Consommateur[]
 - IdDemandeRoutage
 - Identifiant/ Type / Libelle

- FrequenceIndividu / CleRegroupement / PrioRegroupement
- Enrichissement[] : Key / Value
- Resultat
 - Statut
 - Complement[] : Code / Libelle
- Option[] : Key / Value

Selon les étapes, tous les champs ne sont pas forcément renseignés. Les données sont enrichies au fur et à mesure du traitement.

Cette structure est utilisée par les documents suivants :

- sgeCommonsDocs.docs.dm.bpm:StartBpmDm
- sgeCommonsDocs.docs.dm.bpm:StartRouteMsg
- sgeCommonsDocs.docs.dm.bpm:EndRouteMsg
- sgeCommonsDocs.docs.dm.bpm:EndMsg
- sgeCommonsDocs.docs.dm.bpm:EndBpmDm
- sgeCommonsDocs.docs.dm.bpm:CallSNGI
- sgeCommonsDocs.docs.dm.bpm:CallDCR

Ces documents ont tous la même structure :

- IdDemande : identifiant de la demande d'entrée (champ Identifiant de la table DEMANDES)
- IdMessage : identifiant du message d'entrée (champ Identifiant de la table MESSAGES)
- IdMessageTech : identifiant technique du message (champ Id de la tables MESSAGES)
- Entete :
 - structure sgeCommonsDocs.docs.dm.bpm:Data

Note : c'est « IdMessageTech » qui est utilisé pour retrouver le message en BDD (table MESSAGES).

3.5. SgeBpmCommons

Les services webMethods de routage sont regroupés dans le package **SgeBpmCommons**.

3.5.1. Services publics

sgeBpmCommons.sched:reinitCacheRFO

Ce service est programmé à intervalles réguliers (tous les jours). Il réalise la réinitialisation du cache RFO après récupération des risques et domaines, via appel de la méthode [JAVA initRisqueListFromRFO](#)

4. Jars

Les méthodes JAVA de routage sont regroupées dans le jar du package **SgeBpmCommons**.

4.1. CacheRFO.initRisqueListFromRFO

Ce service réalise l'appel, en `remoteInvoke` sur l'IS A2A (sur le premier nœud disponible, via utilisation de la méthode `remoteInvokeMultiNodes`), du service suivant du connecteur RFO :

- `sgeConnectorRFO.pub:getAllRisquesRDomainesValides`

Il est utilisé :

- soit par le service de consultation du cache, lors de la première consultation;
- soit par le service de réinitialisation du cache, à intervalles réguliers (tous les jours).

4.2. CacheRFO.getRisqueListFromRFO

Entrées	<ul style="list-style-type: none">• N/A
Sorties	<ul style="list-style-type: none">• <code>ConcurrentMap<String, List<String>> risqueListFromRFO</code> = liste des risques issus du RFO et des domaines associés à chacun de ces risques.

Ce service, commun aux natures d'échange DF et DM, retourne la liste des risques et domaines du RFO conservés en cache dans une `ConcurrentHashMap`. Si la liste est nulle ou vide, la méthode JAVA [initRisqueListFromRFO](#) est appelée pour l'initialiser avant de la retourner. Si la liste est vide suite à cette initialisation, une exception est levée.

4.3. GenericRoutingManager.routeWithRestriction

Ce service, commun aux natures d'échange DF et DM, implémente l'algorithme fonctionnel de prise en compte de la liste restrictive, décrit dans le document [04]. Selon la nature d'échange, la liste restrictive donnée en paramètre du service est différente (récupérée au niveau du fichier pour DF, ou bien récupérée au niveau du message pour DM).

4.4. GenericRoutingManager.routeWithRisques

Ce service, commun aux natures d'échange DF et DM, implémente l'algorithme fonctionnel de prise en compte de la liste des risques, décrit dans le document [04]. Selon la nature d'échange, la liste des risques donnée en paramètre du service est différente (récupérée au niveau du fichier pour DF, ou bien récupérée au niveau du message pour DM).

4.5. GenericRoutingManager.routeWithComplement

Ce service, commun aux natures d'échange DF et DM, implémente l'algorithme fonctionnel de prise

en compte de la liste complémentaire, décrit dans le document [04]. Selon la nature d'échange, la liste complémentaire donnée en paramètre du service est différente (récupérée au niveau du fichier pour DF, ou bien récupérée au niveau du message pour DM).

4.6. GenericRoutingManager.routeWithAbonnement

Ce service, spécifique à la nature d'échange DM, implémente l'algorithme fonctionnel de prise en compte des abonnements individus, décrit dans les documents [06] et [07]. L'algorithme est identique quel que soit le mode de communication d'entrée utilisé (batch ou temps réel).

4.7. GenericRoutingManager.routeWithDCR

Ce service, spécifique à la nature d'échange DM, implémente l'algorithme fonctionnel de prise en compte des DCR, décrit dans les documents [06] et [07]. L'algorithme est identique quel que soit le mode de communication d'entrée utilisé (batch ou temps réel).

4.8. GenericRoutingManager.getFinalResult

Ce service, commun aux natures d'échange DF et DM, implémente l'algorithme fonctionnel de détermination du résultat final de routage, décrit dans le document [04].

4.9. DfRoutingManager.storeRoutingResult

Ce service, spécifique à la nature d'échange DF, permet de stocker en base de données les demandes de routage pour le fichier courant :

- Autant de demandes de routage que de consommateurs « OK »
- Ou bien, si aucun consommateur « OK », une ligne « demande de routage KO »

4.9.1. Entrées/Sorties

Entrées	<ul style="list-style-type: none">• Demande demande = demande (table ESB.DEMANDES) concernée par le routage• RoutingResult routingResult = résultat de routage• String serviceRef = référence du service d'échange• String fonction = fonction de la demande d'échange courante
Sorties	<ul style="list-style-type: none">• Map<String, String> demandeRoutageIdList = liste des identifiants BDD de la table DEMANDE_ROUTAGES

4.9.2. Algorithme

Si fonction!= 53 (pas de stockage dans le cas « 53 » = « test par le producteur ») :

- Pour chaque consommateur dont le résultat de routage est OK :
 - Insertion d'un tuple dans la table DEMANDE_ROUTAGES pour la demande courante avec :
 - Référence du service
 - Identifiant du consommateur
 - Nature = « DiffusionFichier » / Phase = « DF_SORTIE »
 - Phase = « INITIALISATION » / Etat = « OK »
 - Ajout de l'identifiant du tuple au résultat **demandeRoutageIdList**, pour le consommateur courant
- Si aucun consommateur OK :
 - Insertion d'un tuple dans la table DEMANDE_ROUTAGES pour la demande courante avec :
 - Référence du service
 - <pas d'identifiant consommateur>
 - Nature = « DiffusionFichier » / Phase = « DF_SORTIE »
 - Phase = « INITIALISATION » / Etat = « KO »

4.10. DfRoutingManager.buildEndBpmDfData

Ce service, spécifique à la nature d'échange DF, permet de construire le document EndBpmDf contenant :

- les données d'ARLE à remettre au producteur (décrit dans le document [04])
- les options d'envoi pour la construction des fichiers DF_ARLE et DF_SORTIE

4.10.1. Entrées/Sorties

Entrées	<ul style="list-style-type: none"> • RoutingResult routingResult = résultat de routage • String serviceRef = référence du service d'échange • String exchangeId = identifiant de la demande d'échange courante • String producteurId = identifiant du producteur de l'échange courant • String fonction = fonction de la demande d'échange courante • Map<String, String> demandeRoutageIdList = liste des identifiants BDD de la table DEMANDE_ROUTAGES
Sorties	<ul style="list-style-type: none"> • IData EndBpmDf = contenu du document EndBpmDf à publier

4.10.2. Algorithme

Construction du document EndBpmDf avec les valeurs suivantes :

- EchangeOrigine/Identifiant = <exchangeId>

- ARLE/Statut = « A » si résultat global de routage OK, « R » sinon
- ARLE/Complement= Codes et Libelles du résultat global de routage
- ARLE/Option :
 - Key=SendARLE ⇒ Value= « false » si fonction = 54, « true » sinon
 - Key=ListeConsommateurs ⇒ Value= « true » si option producteur « communiquer la liste des destinataires » = OUI, « false » sinon
- Pour chaque consommateur du résultat de routage :
 - Consommateur/Identifiant = identifiant RFO du consommateur
 - Consommateur/IdDemandeRoutage = si statut OK, identifiant du tuple sauvegardé dans la table DEMANDE_ROUTAGES pour ce consommateur, champ absent sinon
 - Consommateur/Type = « RFO »
 - Consommateur/Libelle = libellé du consommateur issu du catalogue
 - Consommateur/Resultat/Statut = résultat du consommateur (« OK » ou « KO »)
 - Consommateur/Resultat/Complement[]= Codes et Libelles du consommateur
 - Consommateur/Option :
 - Key=SendSORTIE ⇒ Value= « false » si fonction = 53, « true » sinon
 - Key=ListeConsommateurs ⇒ Value=« true » si option service et option consommateur « communiquer la liste des destinataires » = OUI, « false » sinon

4.11. DmRoutingManager.startMsg

Ce service, spécifique à la nature d'échange DM, est la première étape du processus de traitement d'un message, et consiste, pour le message courant, à contrôler les données puis à déclencher l'appel au SNGI si nécessaire ou bien à déclencher l'étape suivante du processus.

La liste détaillée des contrôles et codes d'erreur associés est présente dans les fichiers [03] et [08]. En cas d'échec de l'un des contrôles, les contrôles se poursuivent afin de remonter toutes les erreurs relevées.

4.11.1. Déclenchement

- Sur réception d'un document **StartBpmDm** en provenance du connecteur DM

4.11.2. Entrées/Sorties

Entrées	<ul style="list-style-type: none"> • Doc StartBpmDm contenant les données du message à contrôler
Sorties	<ul style="list-style-type: none"> • Doc CallSNGI ou StartRouteMsg ou EndMsg contenant les données du message enrichies avec : <ul style="list-style-type: none"> ◦ résultat courant des contrôles (statut et codes/libellés)

4.11.3. Algorithme

Contrôles d'entrée

- Mise à jour BDD MESSAGES : Phase = **VALIDATION** / Etat =**ENCOURS**
- Contrôle de l'entête du message
- Contrôle des données métier du message récupérées en BDD MESSAGES, selon options
- Contrôle des balises SNGI du message, selon options SNGI et DCR
- Construction des données à publier :
 - Données reçues en entrée du service + résultat courant des contrôles
- Si au moins un contrôle est KO :
 - Mise à jour BDD MESSAGES : Phase =**VALIDATION** / Etat =**KO**
 - Publication du document **EndMsg** pour déclencher l'étape de fin de traitement
- Sinon :
 - Si un appel au SNGI est nécessaire :
 - Mise à jour BDD MESSAGES : Phase =**VALIDATION** / Etat =**ATT_SNGI**
 - Publication du document **CallSNGI** vers l'IS A2A pour déclencher l'appel au SNGI
 - Sinon :
 - Mise à jour BDD MESSAGES : Phase =**VALIDATION** / Etat =**OK**
 - Publication du document **StartRouteMsg** pour déclencher l'étape de routage

4.12. DmRoutingManager.startRouteMsg

Ce service, spécifique à la nature d'échange DM, est la deuxième étape du processus de traitement d'un message, et consiste, pour le message courant, à contrôler le résultat SNGI si un appel SNGI a eu lieu, à démarrer la détermination des destinataires, puis à déclencher l'appel au DCR si nécessaire ou bien à déclencher l'étape suivante du processus.

4.12.1. Déclenchement

- Sur réception d'un document **StartRouteMsg** en provenance :
 - de l'étape startMsg du processus (si pas d'appel SNGI)
 - ou du connecteur SNGI (réponse SNGI)

4.12.2. Entrées/Sorties

Entrées

- **Doc StartRouteMsg** contenant les données du message enrichies avec :
 - résultat courant des contrôles (statut et codes/libellés)
 - résultat SNGI (si appel SNGI)

Sorties

- **Doc EndMsg ou CallDCR ou EndRouteMsg** contenant les données du message enrichies avec :
 - résultat final des contrôles (statut et codes/libellés), y compris contrôle SNGI
 - résultat courant du routage, par consommateur (statut et codes/libellés)
 - données issues du routage par abonnement individu (enrichissement, regroupement...)

4.12.3. Algorithme

Si déclenché par une réponse SNGI, finalisation des contrôles d'entrée (état courant = ATT_SNGI) :

- Contrôle de la réponse SNGI
- Construction des données à publier :
 - Données reçues en entrée du service + résultat final des contrôles
 - Réponse et résultat SNGI
- Mise à jour BDD MESSAGES : Phase **VALIDATION** / Etat = **OK ou KO**
- Si au moins un contrôle est KO :
 - Publication du document **EndMsg** pour déclencher l'étape de fin de traitement

Détermination des destinataires

- Si tous les contrôles sont OK:
 - Mise à jour BDD MESSAGES: Phase =**ROUTAGE** / Etat =**ENCOURS**
 - Prise en compte de la liste restrictive (1): appel à [routeWithRestriction](#)
 - Prise en compte de la liste des risques(2) : appel à [routeWithRisques](#)
 - Prise en compte des abonnements individus(3) : appel à [routeWithAbonnement](#)
 - Construction des données à publier:
 - Données reçues en entrée du service + résultat courant du routage
 - Si abonnementindividu :
 - clés/valeurs d'enrichissement
 - identifiant et priorité de regroupement
 - fréquence du message
 - Si un appel au DCR est nécessaire:
 - Mise à jour BDD MESSAGES: Phase =**ROUTAGE** / Etat =**ATT_DCR**
 - Publication du document **CallDCR** vers l'IS A2A pour déclencher l'appel au DCR
 - Sinon:
 - Publication du document **EndRouteMsg** pour déclencher l'étape de fin de routage

(1) accès au cache du catalogue, cf. document [05]

(2) accès au cache RFO

(3) accès direct au référentiel des abonnements individus, cf. document [05]

4.13. DmRoutingManager.endRouteMsg

Ce service, spécifique à la nature d'échange DM, est la troisième étape du processus de traitement d'un message, et consiste, pour le message courant, à contrôler le résultat DCR si un appel DCR a eu lieu, à finaliser la détermination des destinataires, puis à déclencher l'étape suivante du processus.

4.13.1. Déclenchement

- Sur réception d'un document **EndRouteMsg** en provenance :
 - de l'étape startRouteMsg du processus (si pas d'appel DCR)
 - ou du connecteur DCR (réponse DCR)

4.13.2. Entrées/Sorties

Entrées	<ul style="list-style-type: none">• Doc EndRouteMsg contenant les données du message enrichies avec :<ul style="list-style-type: none">◦ résultat final des contrôles (statut et codes/libellés), y compris contrôle SNGI◦ résultat courant du routage, par consommateur (statut et codes/libellés)◦ données issues du routage par abonnement individu (enrichissement, regroupement...)◦ résultat DCR (si appel DCR)
Sorties	<ul style="list-style-type: none">• Doc EndMsg contenant les données du message enrichies avec :<ul style="list-style-type: none">◦ résultat final du routage, par consommateur (statut et codes/libellés)

4.13.3. Algorithme

Finalisation de la détermination des destinataires

- Si déclenché par une réponse DCR (état courant = ATT_DCR) :
 - Prise en compte des DCR : appel à [routewithdcr](#)
- Prise en compte de la liste complémentaire(*) : appel à [routeWithComplement](#)
- Détermination du résultat final de routage : appel à [getFinalResult](#)
- Construction des données à publier :
 - Données reçues en entrée du service + résultat final du routage
- Publication du document **EndMsg** pour déclencher l'étape de fin de traitement

(*) accès au cache du catalogue, cf. document [05]

4.14. DmRoutingManager.endMsg

Ce service, spécifique à la nature d'échange DM, est la dernière étape du processus de traitement d'un message, et consiste, pour le message courant, à stocker les demandes de routage en base de données, puis à signaler la fin de traitement du message au connecteur DM en lui transmettant les informations nécessaires à la création des messages CRF et des messages de SORTIE.

4.14.1. Déclenchement

- Sur réception d'un document **EndMsg** en provenance :
 - de l'étape startMsg du processus (si contrôle KO)
 - ou de l'étape startRouteMsg du processus (si contrôle KO)
 - ou de l'étape endRouteMsg du processus

4.14.2. Entrées/Sorties

Entrées	<ul style="list-style-type: none">• Doc EndRouteMsg contenant les données du message enrichies avec :<ul style="list-style-type: none">◦ résultat final des contrôles (statut et codes/libellés), y compris contrôle SNGI◦ résultat final du routage, par consommateur (statut et codes/libellés)
Sorties	<ul style="list-style-type: none">• Doc EndBpmDm contenant les données du message enrichies avec :<ul style="list-style-type: none">◦ options d'envoi vers le producteur et les consommateurs du message

4.14.3. Algorithme

Finalisation des contrôles d'entrée

- Si Phase **ROUTAGE**, stockage des demandes de routage, pour chaque destinataire : appel à [storeRoutingResult](#)
- Mise à jour BDD MESSAGES : Phase **ROUTAGE** / Etat = **OK**, **OK_WARN** ou **KO**
- Construction des données à publier vers le connecteur : appel à [buildEndBpmDmData](#)
- Publication du document **EndBpmDm** pour signaler la fin du processus

4.15. DmRoutingManager.storeRoutingResult

Ce service, spécifique à la nature d'échange DM, permet de stocker en base de données les demandes de routage pour le message courant :

- Autant de demandes de routage que de consommateurs « OK »
- Ou bien, si aucun consommateur « OK », une ligne « demande de routage KO »

4.15.1. Entrées/Sorties

Entrées	<ul style="list-style-type: none">• String id = identifiant du message concerné par le routage (table ESB.MESSAGES)• RoutingResult routingResult = résultat de routage• String serviceRef = référence du service d'échange• String fonction = fonction de la demande d'échange courante
Sorties	<ul style="list-style-type: none">• Map<String, String> messageRoutageIdList = liste des identifiants BDD de la table MESSAGE_ROUTAGES

4.15.2. Algorithme

Enrichissement et stockage du message à router

Si fonction != 53 (pas de stockage dans le cas « 53 » = « test par le producteur ») :

- Pour chaque consommateur dont le résultat de routage est OK :
 - Création d'un tuple MESSAGE_ROUTAGES en BDD pour le message courant avec :
 - Référence du service
 - Identifiant du consommateur
 - Nature = « DiffusionMessages » / Phase = « DM_SORTIE »
 - Phase = « INITIALISATION » / Etat = « OK »
 - Identifiant de regroupement = si abonnement, issu de l'abonnement, sinon, du service
 - Priorité de regroupement = si abonnement, issu de l'abonnement
 - Date d'envoi prévue = si abonnement, calculé avec fréquence abonnement individu
 - Ajout de l'identifiant du tuple au résultat **messageRoutageIdList**, pour le consommateur courant
- Si aucun consommateur OK :
 - Création d'un tuple MESSAGE_ROUTAGES en BDD pour le message courant avec :
 - Référence du service
 - <pas d'identifiant consommateur>
 - Nature = « DiffusionMessages » / Phase = «DM_SORTIE»
 - Phase = « INITIALISATION » / Etat = « KO »

4.16. DmRoutingManager.buildEndBpmDmData

Ce service, spécifique à la nature d'échange DM, permet de construire le document EndBpmDm contenant :

- les données de CRF à remettre au producteur (décrit dans le document [03])
- les options d'envoi pour la construction des messages de CRF, ainsi que des messages de sortie à destination de chaque consommateur.

4.16.1. Entrées/Sorties

Entrées	<ul style="list-style-type: none"> • Doc enteteEndMsg contenant les données de l'entete du message • String serviceRef = référence du service d'échange • String producteurId = identifiant du producteur de l'échange courant • String fonction = fonction de la demande d'échange courante • Map<String, String> messageRoutageIdList = liste des identifiants BDD de la table MESSAGE_ROUTAGES
Sorties	<ul style="list-style-type: none"> • IData EndBpmDm = contenu du document EndBpmDm à publier

4.16.2. Algorithme

Construction du document EndBpmDm avec les valeurs suivantes :

- Données du message d'entrée (hors listes de routage, clés et valeurs de filtrage)
- Données d'enrichissement du message :
 - Clés et valeurs d'enrichissement
 - Résultat et réponse SNGI, si un appel SNGI a eu lieu
- Données producteur :
 - Resultat/Statut = « OK », « OK avec signalement » ou « KO »
 - Resultat/Complement[] = Codes et Libellés de validation et de routage
 - Options :
 - Key=Send ⇒ Value=« false » si fonction = 54, « true » sinon
 - Key=Consommateurs ⇒ Value=« true » si option producteur « communiquer la liste des destinataires » = OUI, « false » sinon
 - Key=SNGI ⇒ Value=« true » si option producteur « communiquer le résultat d'identification SNGI » = OUI, « false » sinon
- *Pour chaque consommateur,*
 - Données consommateur :
 - Identifiant = identifiant RFO du consommateur
 - IdDemandeRoutage = si statut OK, identifiant du tuple sauvegardé dans la table MESSAGE_ROUTAGES pour ce consommateur, champ absent sinon
 - Type = « RFO »
 - Libelle = libellé du consommateur issu du catalogue

- Resultat/Statut = résultat du consommateur (« OK » ou « KO »)
- Resultat/Complement[]= Codes et Libelles du consommateur
- Options :
 - Key=Send ⇒ Value=« false » si fonction = 53, « true » sinon
 - Key=Consommateurs ⇒ Value=« true » si option service et option consommateur « communiquer la liste des destinataires » = OUI, « false » sinon
 - Key=SNGI ⇒ Value=« true » si option consommateur « communiquer le résultat d'identification SNGI » = OUI, « false » sinon

5. Annexes

5.1. Marques de révision

Différences depuis le dernier tag

