|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | Projet BI |
|  | |  |  |
|  |  |  |  |
| **Version :** | *V10* |  |  |
| **Date :** | *12/02/2018* |  |  |
|  |  |  |  |
|  |  |  | Définition du Framework Talend |

|  |  |
| --- | --- |
|  | |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

| **Historique des versions** | | |
| --- | --- | --- |
| **N° Version** | **Date** | **Description** |
| 0.1 | 27/12/2017 | Initialisation du document |
| 0.2 | 04/01/2018 | Relecture |
| 0.3 | 08/01/2018 | Ajout de la partie chargement ODS et DWH |
| 0.4 | 16/01/2018 | Modification des logs |
| 0.5 | 06/02/2018 | Mise à jour du Framework |
| 0.9 | 12/02/2018 | Relecture Aliasir |

Table des matières

[Objet du document 3](#_Toc506297350)

[1 Traitements Transverses 4](#_Toc506297351)

[1. Fonctionnement du système de log 4](#_Toc506297352)

[1.1.1 Log de chaîne (globale) 4](#_Toc506297353)

[1.1.2 Log de Traitement - ETAPE 5](#_Toc506297354)

[1.1.3 Log d’exécution - DETAILS 6](#_Toc506297355)

[2. Description détaillée des Tables de logs : 8](#_Toc506297356)

[1.1.4 Table d’exécution de la chaîne - ADMLOG\_EXECUTION\_CHAINE 8](#_Toc506297357)

[1.1.5 Table de référence des traitements - ADMLOG\_REFERENTIEL\_TRAITEMENT 9](#_Toc506297358)

[1.1.6 Table Exécution des traitements - ADMLOG\_EXECUTION\_TRAITEMENT 10](#_Toc506297359)

[1.1.7 Table de log d’exécution - ADMLOG\_EXECUTION\_LOG 11](#_Toc506297360)

[1.1.8 Table des types de log d’exécution - ADMLOG\_TYPLOG 12](#_Toc506297361)

[1.1.9 Table Référentiel des messages d'erreur - ADMLOG\_REFERR 13](#_Toc506297362)

[3. Description de l’Implémentation du système de log dans les traitements 14](#_Toc506297363)

[4. Gestion de l’archivage et du rejet des lots de fichiers 15](#_Toc506297364)

[2 cinematique de l’application 16](#_Toc506297365)

[1. Démarrage de l’application 16](#_Toc506297366)

[2. Démarrage du traitement 16](#_Toc506297367)

[2.1.1 Test & chargement du contexte 16](#_Toc506297368)

[2.1.2 Test connexion à la source et la cible de données 16](#_Toc506297369)

[3. Phase 0 : Démarrage effectif de l’application 17](#_Toc506297370)

[4. Phase 1 : Connexion aux sources de données 17](#_Toc506297371)

[5. Phase 2 : Vérification des fichiers PAR SOURCE 18](#_Toc506297372)

[2.1.3 Phase 3B : Transfert des fichiers de SAS vers Dossier « travail » 18](#_Toc506297373)

[6. Alimentation 19](#_Toc506297374)

[7. Fin d’une action 19](#_Toc506297375)

[8. Fin de traitement 19](#_Toc506297376)

[9. Fin de la chaîne 19](#_Toc506297377)

[3 Traitements Staging 20](#_Toc506297378)

[10. Phase 1 : STAGING brutes 20](#_Toc506297379)

[3.1.1 STAGING brutes : Initialisation de la log 20](#_Toc506297380)

[3.1.2 STAGING brutes : Vidage 20](#_Toc506297381)

[3.1.3 STAGING brutes : contrôle niveau 1 20](#_Toc506297382)

[3.1.4 STAGING brutes : Chargement 21](#_Toc506297383)

[3.1.5 STAGING brutes : contrôle niveau 2 21](#_Toc506297384)

[3.1.6 STAGING brutes : contrôle niveau 3 22](#_Toc506297385)

[3.1.7 Contrôle des données des tables STAGING brut 22](#_Toc506297386)

[Phase 2 : Chargement des tables STAGING final 22](#_Toc506297387)

[11. Phase 4 : fin du job d’extraction des fichiers sources et archivage 23](#_Toc506297388)

[4 Traitements ODS 24](#_Toc506297389)

[12. Phase 1 : Démarrage du job de chargement de l’ODS 24](#_Toc506297390)

[4.1.1 Etape5 : Démarrage du job de chargement de ODS 24](#_Toc506297391)

[4.1.2 Etape6 : Chargement de ODS 24](#_Toc506297392)

[4.1.3 Etape6 : Mise à jour de la log 24](#_Toc506297393)

[5 Traitements DWH 25](#_Toc506297394)

[13. Phase 1 : Démarrage du job de chargement de DWH. 25](#_Toc506297395)

[6 Annexes usage des flags et commit 26](#_Toc506297396)

Objet du document

L’objet de ce document est de présenter la mise en place du Framework du projet BI XXXXXX qui permettra aux équipes projet la mutualisation et la réutilisation des jobs Talend afin d’accélérer les développements.

Le Framework désigne un ensemble cohérent de composants, qui va servir à créer les fondations ainsi que les grandes lignes de l’application XXXXXX BI.

Ce document comprend les éléments suivants :

* **Traitements transverses**

Cette partie est dédiée aux traitements standards qui ont donc été rationalisés en vue d'une utilisation "multiple" au travers de différents jobs.

Ainsi, les logs, la gestion des erreurs et le système d’archivage seront abordés dans cette partie.

* **Cinématique de l’application**

Présentation globale du système, depuis le démarrage, de la chaîne d’exécution des traitements.

Il s'agira du "squelette" de l'application.

* **Traitements "STAGING"**

Cette partie sera consacrée aux traitements à effectuer sur la zone contenant les tables de travail et de rejets.

* **Traitements ODS**

Cette partie sera consacrée à l’alimentation des tables du modèle métier.

* **Traitements DWH**

Cette partie sera consacrée à l'alimentation de l'espace final contenant les tables de dimensions et faits.

# Traitements Transverses

Le suivi et la surveillance au travers d'un système de log sont les premières des nécessités techniques de l’application.

Il s’agira donc de :

* Tracer les différents opérations/traitements dans une base de données en vue de rapports et éventuellement alertes
* Afficher l'information d'étapes dans la console d'exécution.

## Fonctionnement du système de log

3 tables sont alimentées lors de l’exécution d’une chaîne :

1. La table d’exécution de la chaîne : ADMLOG\_EXECUTION\_CHAINE
2. La table d’exécution des traitements : ADMLOG\_EXECUTION\_TRAITEMENT
3. La table des logs d’exécution : ADMLOG\_EXECUTION\_LOG

### Log de chaîne (globale)

**Au lancement de la chaîne**, un identifiant d’exécution de chaîne est obtenu en **AUTOINCREMENT**.

L’identifiant d’exécution sera donc référencé dans :

* une nouvelle ligne dans la table ADMLOG\_EXECUTION\_CHAINE
* le champ ADMLOGEXECHN\_ID
  + Associé à
    - ADMLOGEXECHN\_DEB\_DAT : date de début de lancement de la chaîne
    - ADMLOGEXECHN\_ETA : A l’état 2 (Exécution en cours)

**Lorsque la chaîne**

* **Se termine avec succès** :
* la date de fin de chaîne - ADMLOGEXECHN\_FIN\_DAT - est mise à jour.
* le flag de l’état est mis à jour à « 0 » - « Exécution terminée avec succès »
* **s’arrête pour cause d’erreur** :
* le flag de l’état est consigné à « 1 » - « Exécution en erreur »

**NB : le redémarrage suite à une erreur correspondra à une nouvelle chaîne**

**Table ADMLOG\_EXECUTION\_CHAINE**

|  |  |  |  |
| --- | --- | --- | --- |
| **ETAPE** | CHAMP ID | CHAMP DATE | CHAMP ETAT |
| **LANCEMENT CHAÎNE** |  |  |  |
|  | ADMLOGEXECHN\_ID = AutoIncrement | ADMLOGEXECHN\_DEB\_DAT | ADMLOGEXECHN\_ETA=2 |
|  |  |  |  |
| **FIN CHAÏNE OK** |  | ADMLOGEXECHN\_FIN\_DAT | ADMLOGEXECHN\_ETA=0 |
|  |  |  |  |
| **ERREUR CHAINE** |  | ADMLOGEXECHN\_FIN\_DAT | ADMLOGEXECHN\_ETA=1 |

Une chaîne est composée de plusieurs traitements à lancer en série et/ou en parallèle.

Les types de traitements sont :

* Chargement
* Contrôle
* Transformation

### Log de Traitement - ETAPE

#### Présentation Générale

**Au lancement de chaque traitement**, un identifiant d’exécution du traitement est obtenu en **AUTOINCREMENT**.

Pour permettre le suivi de chaque nouveau traitement :

* une ligne est insérée dans la table ADMLOG\_EXECUTION\_TRAITEMENT
* un identifiant est assigné via le champ ADMLOGEXETRT\_ID.

On retrouvera donc dans cette ligne les informations suivantes :

* L’identifiant du TYPE traitement de référence = champ ADMLOGEXETRT\_REFTRT\_ID
* L’identifiant de la chaîne l’ayant lancé = champ ADMLOGEXETRT\_EXECHN\_ID
* La date de début de lancement du traitement = champ ADMLOGEXETRT\_DEB\_DAT
* L’état du traitement référencé à « 2 » - « Exécution en cours » = champ ADMLOGEXETRT\_ETA.

Ainsi, lorsque

* **le traitement est terminé** (avec succès):
* la date de fin - ADMLOGEXETRT\_FIN\_DAT- est modifiée avec la date de fin de traitement
* l’état/ le flag est mis à jour à « 0 » - « Exécution terminée avec succès »
* **le traitement est arrêté** :
* l’état/ le flag est mise à jour à « 1 » - « Exécution en erreur ».

**NB : le redémarrage suite à une erreur correspondra à un nouveau traitement.**

**Table ADMLOG\_EXECUTION\_TRAITEMENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ETAPE** | **Champ REF CHAINE** | CHAMP ID | CHAMP DATE | CHAMP ETAT |
| **LANCEMENT** | ***ADMLOG\_EXECUTION\_CHAINE*** .ADMLOGEXECHN\_ID | ADMLOGEXECHN\_ID = AutoIncrement | ADMLOGEXETRT\_DEB\_DAT | ADMLOGEXECHN\_ETA=2 |
|  |  |  |  |  |
| **FIN OK** |  |  | ADMLOGEXECHN\_FIN\_DAT | ADMLOGEXECHN\_ETA=0 |
|  |  |  |  |  |
| **ERREUR** |  |  | ADMLOGEXECHN\_FIN\_DAT | ADMLOGEXECHN\_ETA=1 |

#### Liste des étapes

Table

### Log d’exécution - DETAILS

#### Présentation Générale

Lorsqu’un traitement est lancé, **chaque action** est mentionnée dans la table des logs d’exécution « ADMLOG\_EXECUTION\_LOG ».

Dans cette table, le suivi de l’exécution du traitement est borné par de 2 messages dans :

* Un message de début indiquant « DEBUT DU TRAITEMENT xxxxxxxxx » associé sur la même ligne à un ID = <identifiant d’exécution du traitement>;
* Un message de fin indiquant « FIN DU TRAITEMENT IDxxxxxxxxx AVEC SUCCES associé sur la même ligne à un ID = <identifiant d’exécution du traitement>

**NB :** L’identifiant d’exécution du traitement est le même que celui utilisé dans la table de suivi des traitements ADMLOG\_EXECUTION\_TRAITEMENT.

Toutes les lignes de la table des logs d’exécution - *« ADMLOG\_EXECUTION\_LOG »* - sont identifiées/référencées dans le champ **ADMLOGEXELOG\_ID**.

Ces informations sont complétées par des messages fonctionnels - ADMLOGEXELOG\_FCT\_TX - et techniques - ADMLOGEXELOG\_TCH\_TX - de :

* la date et de l’heure du message ADMLOGEXELOG\_DT
* du type de message de log ADMLOGEXELOG\_TYPMSG\_ID

Les types de message de log sont référencés dans la table ADMLOG\_TYPLOG.

**Table ADMLOG\_EXECUTION\_LOG**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ETAPE** | **Champ REF CHAINE** | **CHAMP ID** | **CHAMP DATE** | **MESSAGE LOG** | **TYPE MESSAGE** | **Desc fonctionnel** | **Desc Technique** |
| **LANCEMENT** | ***ADMLOG\_EXECUTION\_CHAINE*** .ADMLOGEXECHN\_ID | ADMLOGEXECHN\_ID  = AutoIncrement | ADMLOGEXELOG\_DT | « DEBUT DU TRAITEMENT xxxxxxxxx »  associé sur la même ligne à un ID  = <identifiant d’exécution du traitement> | ADMLOGEXELOG\_TYPMSG\_ID | ADMLOGEXELOG\_  FCT\_TX | ADMLOGEXELOG\_  TCH\_TX |
|  |  |  |  |  |  |  |  |
| **FIN OK** |  |  |  | « FIN DU TRAITEMENT IDxxxxxxxxx  AVEC SUCCES associé sur la même ligne à un ID  = <identifiant d’exécution du traitement> |  |  |  |
|  |  |  |  |  |  |  |  |
| **ERREUR** |  |  |  | « FIN DU TRAITEMENT IDxxxxxxxxx  AVEC SUCCES associé sur la même ligne à un ID  = <identifiant d’exécution du traitement> |  |  |  |

#### Liste des TRAITEMENTS détaillés

#### Usage transverse de l’identifiant de traitement

L’identifiant d’exécution est également utilisé dans toutes les tables de la solution BI.

Ainsi, toutes les lignes de chaque table contiennent 2 champs :

1. *<préfixe\_de\_la\_table>\_***CRETRT\_ID** : identifiant du traitement ayant créé la ligne
2. *<préfixe\_de\_la\_table>\_***MAJTRT\_ID** : identifiant du dernier traitement ayant modifié la ligne.

#### Type de message de log

##### Les types d’Anomalies / Rejets

On distingue 2 types anomalies lors des traitements qui, par conséquent, génèrent 2 types de rejets :

1. «**Technique**» : liés à une anomalie dans la **structure du fichier ou la nature des données**
2. « **Fonctionnel »** générés suite à la vérification d’une règle de gestion « **métier** » ou un **contrôle d’intégrité référentielle**

.

**Tous ces contrôles sont donc effectués avant l’écriture dans la table cible**.

##### Méthodologie

Compte tenu des différents types d’anomalies et critères de règles de contrôle, le flux doit être contrôlé avec un filtre reposant sur une ou plusieurs conditions.

En sortie des contrôles, les données non satisfaisantes devront être redirigées vers une table de logs (voir le paragraphe 1.2.1)

Pour cela, il convient d’utiliser le composant « **tMap** » avec une sortie qui contient le filtre de conditions

Et/ou une sortie avec le filtre inverse.

Si un problème est rencontré lors des traitements, l’information est donc enregistrée dans la table dédiée et en utilisant des codes prédéfinis via le composant « **tDie »**.

**Table des codes erreurs ADMLOG\_LOG\_REFERR**

|  |  |
| --- | --- |
| Code erreur | Description |
| **0** | Exécution OK |
| **1 à 1000** | Erreurs techniques sur/dans les fichiers (fichier inexistant, lecture, format du fichier, format des données dans le fichier,...) |
| **1001 à 2000** | Erreurs de base de données (connexion impossible etc…) |
| **2001 à 3000** | Erreurs réseau (le web service ne répond pas, connexion à FTP impossible.) |
| **3001 et suite** | Codes personnalisés pour les erreurs fonctionnelles |

Lorsqu’une erreur est détectée lors d’un contrôle, un message est généré sur la base des modèles de la table **ADMLOG\_LOG\_REFERR** avec comme type de message de log « 3 » - «  Erreur ».

Lorsqu’une table est modifiée/mise à jour, un message technique est inséré dans la table au format :

<Nom\_de\_la\_table>;<nb\_lignes\_insérées>;<nb\_lignes\_mises\_à\_jour>;<nb\_de\_lignes\_supprimées>

Le message fonctionnel équivalent est :

« Entité fonctionnelle concernée : <nb> créées, <nb> modifiées et <nb> supprimées ».

## Description détaillée des Tables de logs :

### Table d’exécution de la chaîne - ADMLOG\_EXECUTION\_CHAINE

**Objectif :** Suivi d'exécution GLOBAL avec la date et l’heure de lancement de la chaîne.

Cette chaîne sera référencée dans un identifiant repris dans :

* Le suivi d’exécution
* Le suivi des traitements
* Les logs

APPLI. SOURCE : wombat DB (schéma admin)

APPLI. CIBLE : wombat DB (schéma admin)

|  |
| --- |
| Etape0 Etape 1 |

**Etape 0 :**

Initialisation du log de début d'une exécution (table ADMLOG\_EXECUTION\_CHAINE) dans le job INIT\_ADMLOG\_EXECUTION\_CHAINE

Requête à utiliser :

|  |
| --- |
| "INSERT INTO admin.ADMLOG\_EXECUTION\_CHAINE  (ADMLOGEXECHN\_DEB\_DT, ADMLOGEXECHN\_ETA)  VALUES (CURRENT\_TIMESTAMP, 2);  COMMIT;" |

**Table ADMLOG\_EXECUTION\_CHAINE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | Libellé | Type | Obligatoire | Modèle |
| ADMLOGEXECHN\_ID | Id de l'exécution de la chaîne | int | Oui |  |
| ADMLOGEXECHN\_DEB\_DT | Date heure début exécution | datetime | Oui | "yyyy-MM-dd HH:mm:ss" |
| ADMLOGEXECHN\_FIN\_DT | Date heure fin exécution | datetime | Non | "yyyy-MM-dd HH:mm:ss" |
| ADMLOGEXECHN\_ETA | État | int | Oui |  |

### Table de référence des traitements - ADMLOG\_REFERENTIEL\_TRAITEMENT

**Objectif :** Référencer l’ensemble des anomalies techniques et fonctionnelles

**Description détaillée de la Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | Libellé | Type | Obligatoire | Modèle |
| ADMLOGREFTRT\_ID | ID du traitement | int | Oui |  |
| ADMLOGREFTRT\_TCH\_TX | Libellé technique du traitement | Varchar | Non |  |
| ADMLOGREFTRT\_FCT\_TX | Libellé fonctionnel du traitement | Varchar | Non |  |

Chaque type de traitement est référencé via l’id ADMLOGREFTRT\_ID et possède les descriptifs :

* technique ADMLOGREFTRT\_TCH\_TX
* fonctionnel ADMLOGREFTRT\_FCT\_TX

**Alimentation**

Cette table est alimentée (initialisé/enrichie) **manuellement via les éléments ci-dessous :**

APPLI. SOURCE : wombat DB (schéma admin)

APPLI. CIBLE : wombat DB (schéma admin)

|  |
| --- |
| Etape0 Etape1 |

**Etape 0**

* connexion à l’application via le schéma technique de monitoring

**Etape 1**

* Requête à utiliser pour insérer les données :

|  |
| --- |
| INSERT INTO [admin].[ADMLOG\_REFERENTIEL\_TRAITEMENT]  ([ADMLOGREFTRT\_TCH\_TX]  ,[ADMLOGREFTRT\_FCT\_TX])  VALUES  (<ADMLOGREFTRT\_TCH\_TX, varchar(max),>  ,<ADMLOGREFTRT\_FCT\_TX, varchar(max),>) |

### Table Exécution des traitements - ADMLOG\_EXECUTION\_TRAITEMENT

**Objectif :** suivi d'exécution **UNITAIRE** avec la date et l’heure de lancement du traitement.

**Description détaillée de la Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | Libellé | Type | Obligatoire | Modèle |
| ADMLOGEXETRT\_ID | ID de l'exécution du traitement | int | Oui |  |
| ADMLOGEXETRT\_EXECHN | ID de l'exécution de la chaîne | int | Oui |  |
| ADMLOGEXETRT\_REFTRT | ID du traitement | int | Oui |  |
| ADMLOGEXETRT\_DEB\_DT | Date heure début exécution | datetime | Oui | "yyyy-MM-dd HH:mm:ss" |
| ADMLOGEXETRT\_FIN\_DT | Date heure fin exécution | datetime | Non | "yyyy-MM-dd HH:mm:ss" |
| ADMLOGEXETRT\_ETA | État | int | Oui |  |

**Alimentation**

APPLI. SOURCE : wombat DB - schéma admin

APPLI. CIBLE : wombat DB - schéma admin

|  |
| --- |
| Etape 0 Etape1 Etape2 |

**Etape 0 :**

* Connexion à l’application via le schéma technique de monitoring

**Etape 1**

* Requête

|  |
| --- |
| "INSERT INTO admin.ADMLOG\_EXECUTION\_TRAITEMENT(ADMLOGEXETRT\_EXECHN\_ID,ADMLOGEXETRT\_REFTRT\_ID,ADMLOGEXETRT\_DEB\_DT,ADMLOGEXETRT\_ETA)  OUTPUT INSERTED.ADMLOGEXETRT\_ID  VALUES (IDENT\_CURRENT('admin.ADMLOG\_EXECUTION\_CHAINE'),1,CURRENT\_TIMESTAMP,2);  COMMIT;" |

**Etape 2 :**

* Enregistrement de l’information dans le buffer.

### Table de log d’exécution - ADMLOG\_EXECUTION\_LOG

**Objectif :** Collecter tous les messages nécessaires pour suivre l’exécution détaillée de chaque traitement.

**Description détaillée de la table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | Libellé | Type | Obligatoire | Modèle |
| ADMLOGEXELOG\_ID | ID de log d'exécution | int | Oui |  |
| ADMLOGEXELOG\_LOGTRT\_ID | IID d'exécution du traitement | int | Oui |  |
| ADMLOGEXELOG\_TYPMSG\_ID | Identifiant du type de message de log | int | Oui |  |
| ADMLOGEXELOG\_TCH\_TX | Message technique | String | Oui |  |
| ADMLOGEXELOG\_FCT\_TX | Message fonctionnel | String | Non |  |
| ADMLOGEXELOG\_DT | Date et heure du message | Datetime | Oui | "yyyy-MM-dd HH:mm:ss" |

Chaque ligne correspond à un message identifié/référencé par ADMLOGEXELOG\_ID.

L’information est :

* datée = ADMLOGEXELOG\_DT
* décrite dans un message fonctionnel = ADMLOGEXELOG\_FCT\_TX
* décrite dans un message technique = ADMLOGEXELOG\_TCH\_TX

Chaque information est liée à :

* une exécution de traitement = ADMLOGEXELOG\_LOGTRT\_ID
* un type de message = ADMLOGEXELOG\_TYPMES\_ID

Les messages d’erreur se basent sur les canevas définis dans la table de référence des erreurs ADMLOG\_LOG\_REFERR.

**Alimentation :**

APPLI. SOURCE : wombat DB (schema admin)

APPLI. CIBLE : wombat DB (schema admin)

|  |
| --- |
| Etape 0 Etape1 Etape2 |

**Etape 0**

* Connexion à l’application via le schéma technique de monitoring

**Etape 1**

* Requête

|  |
| --- |
| " INSERT INTO admin. ADMLOG\_ADMLOG\_EXECUTION\_CHAINE (ADMLOGEXELOG\_LOGTRT\_ID, ADMLOGEXELOG\_TYPMSG\_ID, ADMLOGEXELOG\_TCH\_TX, ADMLOGEXELOG\_FCT\_TX, ADMLOGEXELOG\_DT)  OUTPUT INSERTED. ADMLOG\_ADMLOG\_EXECUTION\_CHAINE  VALUES (" + context. ADMLOGEXELOG\_LOGTRT\_ID + ", '" + context. ADMLOGEXELOG\_TYPMSG\_ID + "', '" + context.ADMLOGEXELOG\_TCH\_TX + "', '" + context.ADMLOGEXELOG\_FCT\_TX + "', CURRENT\_TIMESTAMP);  COMMIT;" |

Etape 2

* Enregistrement de l’information dans le buffer.

### Table des types de log d’exécution - ADMLOG\_TYPLOG

**Objectif**: Référencer tous les types de message de log.

**Description détaillée de la Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | Libellé | Type | Obligatoire | Modèle |
| ADMLOGTYPLOG\_ID | ID du type de message de log | int | Oui |  |
| ADMLOGTYPLOG\_TX | Libellé du type de message de log | String | Oui |  |

Chaque type est identifié via le champ ADMLOGTYPLOG\_I et possède un descriptif via le champ ADMLOGTYPLOG\_TX.

Les types de messages sont les suivants :

|  |  |
| --- | --- |
| **Identifiant du type** | **Libellé du type** |
| 1 | Information libre |
| 2 | Métrique |
| 3 | Erreur |

**Alimentation :**

**Cette table est alimentée manuellement**

APPLI. SOURCE : wombat DB (schema admin)

APPLI. CIBLE : wombat DB (schema admin)

|  |
| --- |
| Etape0 Etape1 |

**Etape 0**

* connexion à l’application via le schéma technique de monitoring

**Etape 1**

* Requête

|  |
| --- |
| INSERT INTO [admin].[ADMLOG\_TYPLOG]  ([ADMLOGTYPLOG\_TX])  VALUES  (<ADMLOGTYPLOG\_TX, varchar(max),>) |

### Table Référentiel des messages d'erreur - ADMLOG\_REFERR

Objectif : Référencer l’ensemble des messages d’erreur.

**Description détaillée de la Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | Libellé | Type | Obligatoire | Modèle |
| ADMLOGREFERR\_ID | ID du message d'erreur | int | Oui |  |
| ADMLOGREFERR\_RGE\_ID | ID de la règle de gestion liée à l'erreur | int | Oui |  |
| ADMLOGREFERR\_TCH\_TX | Message technique d'erreur | string | Oui |  |
| ADMLOGREFERR\_FCT\_TX | Message fonctionnel d'erreur | string | Oui |  |

Un message est identifié par le champ ADMLOGREFERR\_ID et associé à :

* L’identifiant de la règle de gestion - ADMLOGREFERR\_REG\_ID
* Le message technique de l’erreur - ADMLOGREFERR\_TCH\_TX
* Le message fonctionnel de l’erreur - ADMLOGREFERR\_FCT\_TX

Les messages sont des modèles contenant des chaînes de caractère « **%Mn% »** à remplacer lors de l’insertion des messages dans la table des logs d’exécution.

**Alimentation :**

**Cette table est alimentée manuellement.**

APPLI. SOURCE : wombat DB (schema admin)

APPLI. CIBLE : wombat DB (schema admin)

|  |
| --- |
| Etape0 Etape1 |

**Etape 0**

* Connexion à l’application via le schéma technique de monitoring

**Etape 1**

* Requête de mise à jour

|  |
| --- |
| INSERT INTO [admin].[ADMLOG\_REFERR]  ([ADMLOGREFERR\_RGE\_ID]  ,[ADMLOGREFERR\_TCH\_TX]  ,[ADMLOGREFERR\_FCT\_TX])  VALUES  (<ADMLOGREFERR\_RGE\_ID, numeric(18,0),>  ,<ADMLOGREFERR\_TCH\_TX, varchar(max),>  ,<ADMLOGREFERR\_FCT\_TX, varchar(max),>) |

## Description de l’Implémentation du système de log dans les traitements

L’implémentation des logs est gérée via le joblet « **JOBLET\_logging** ».

Celui-ci, placé dans le job à suivre, se charge de l'écriture dans le fichier et de la règle de nommage des fichiers.

**NB :** Dans un job Talend, toute donnée placée dans un composant « **tWarn »** est écoutée et écrite dans la table et/ou fichier.

L'ensemble est illustré dans la figure ci-après :

|  |
| --- |
|  |

## Gestion de l’archivage et du rejet des lots de fichiers

L’archivage vise essentiellement les fichiers sources.

Si les traitements jusqu’au contrôle de niveau 3 se sont déroulés correctement, l’archivage est déclenché après le chargement de la STAGING final (voir phase2).

Par conséquent, les fichiers du lot concerné -SBO, TWISTER, NXDENSITY, STANDARD ou TRANSCO- sont zippés et déplacés dans le répertoire **ARCHIVE**.

Si une anomalie est détectée lors des contrôles de niveau 0 à niveau 3, les fichiers du lot concerné -SBO, TWISTER, NXDENSITY, STANDARD ou TRANSCO- sont zippés et déplacés dans le répertoire de **REJET**.

Le déplacement se fera donc –en fonction - à la fin des contrôles de niveau 0, 1et 2, ou 3

|  |
| --- |
|  |

En cas d’erreur, celle-ci sera reportée dans la table des erreurs avec un suivi des logs conformément aux paragraphes dédiés.

On distinguera donc les répertoires d’archive :

* du lot dont au moins un fichier a rencontré au moins une erreur
  + BI\**Archivage**
* du lot dont l’alimentation s’est passé correctement
  + BI\**Rejet**

# cinematique de l’application

Cette section présente la logique d’enchaînements des différentes étapes dans l’application.

## Démarrage de l’application

En préambule du lancement effectif de l’application, l’exécution de la chaîne est initialisée dans la table de log ADMLOG\_EXECUTION\_CHAINE (Cf. le paragraphe 1.2.1.1)

## Démarrage du traitement

Le composant « **tPrejob »** démarre l'exécution d'un pré-job.

### Test & chargement du contexte

Au lancement de l’application, vérifier que les variables de contextes nécessaires ont bien été ajoutées dans ce job avant le démarrage :

|  |
| --- |
|  |

Un JOBLET sera utilisé pour factoriser ce traitement nommé : **LOAD\_CONTEXT.**

En cas d’erreur, celle-ci sera reportée dans la table des erreurs avec un suivi des logs conformément aux paragraphes dédiés.

### Test connexion à la source et la cible de données

Ouverture en parallèle des connexions vers les sources et cibles de données :

|  |
| --- |
|  |

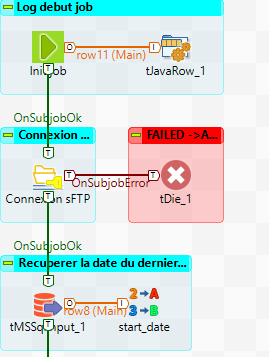
Tester les connexions en simultané /parallèle :

* Si les 2 sont OK : on continue
* Si au moins une des 2 est KO alors ERREUR
  + L’erreur reportée dans la table des erreurs avec un suivi des logs conformément aux paragraphes dédiés.

## Phase 0 : Démarrage effectif de l’application

Initialisation du log de début d'un traitement (table ADMLOG\_EXECUTION\_TRAITEMENT) dans le job **INIT\_ADMLOG\_EXECUTION\_TRAITEMENT** : (Voir le paragraphe log 1.2.1.2)

## Phase 1 : Connexion aux sources de données



**Etape 1**

* Stocker l’id ADMLOG\_EXECUTION\_TRAITEMENT du démarrage de l’application

**Etape 2**

* Se connecter au répertoire distant pour accéder aux fichiers sources
  + Si l’accès au répertoire est impossible – Étape 2A -> Stop (composant utilisé est « **TDIE »**)

**Etape 3** (l’accès au répertoire est OK)

* Récupérer la date de la dernière exécution réussie.

Cette date sera nommée "**start\_date**" via la requête ci-dessous et contrôlée dans la phase suivante.

Requête à utiliser :

|  |
| --- |
| *"SELECT CAST(COALESCE(DATEADD(day,1, MAX(ADMLOGEXECHN\_DEB\_DT)), CURRENT\_TIMESTAMP) AS DATE) AS start\_date*  *FROM admin.ADMLOG\_EXECUTION\_CHAINE*  *WHERE Etat = 0"* |

**NB : En cas d’erreur, celle-ci sera reportée dans la table des erreurs avec un suivi des logs conformément aux paragraphes dédiés.**

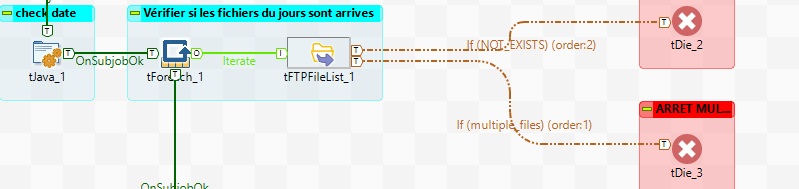
## Phase 2 : Vérification des fichiers PAR SOURCE

### Phase 3B : Transfert des fichiers de SAS vers Dossier « travail »

Envoi des fichiers récupérés de SAS au dossier travail

|  |
| --- |
|  |

NB : En cas d’erreur, celle-ci sera reportée dans la table des erreurs avec un suivi des logs conformément aux paragraphes dédiés.



**Etape 1 :**

* vérification des traitements sur la base de la date récupérée dans la phase précédente.

**Etape 2 :**

* Vérifier si les fichiers à charger via le composant « **TFOREACH »** :
  + Chaque fichier attendu est présent
  + Chaque fichier n’est présent qu’une fois
* **Etape 2A :**
  + Si un de ces fichiers n’est pas arrivé -> Stop
* **Etape 2B :**
  + Si le fichier du jour est présent plus d'une fois sur le répertoire -> Stop

NB : En cas d’erreur, celle-ci sera reportée dans la table des erreurs avec un suivi des logs conformément aux paragraphes dédiés.

## Alimentation

L’organisation et le découpage des traitements suivent la logique d’alimentation des données dans l’application.

Ainsi, les traitements suivront l’alimentation des tables et se feront donc dans l’ordre suivant :

1. STAGING Brutes
2. STAGING Finales
3. ODS
4. DWH

*Ces étapes sont décrites de manière détaillée dans les chapitres 3 à 5.*

## Fin d’une action

A la fin d’une tâche, la table **admin.ADMLOG\_EXECUTION\_LOG** est mise à jour.

* **(Voir le paragraphe 1.2.1.3)** pour indiquer que cette tâche ait bien été effectuée ou pas:

Requête mise à jour de la table de log ADMLOG\_EXECUTION\_LOG :

|  |
| --- |
| *" UPDATE admin.ADMLOG*\_*EXECUTION\_LOG*  *SET ADMLOGEXELOG\_DT= CURRENT\_TIMESTAMP, ADMLOGEXELOG\_TCH\_TX= " + ((Integer)globalMap.get("tMSSqlOutput\_1\_NB\_LINE\_INSERTED")) + ",*  *WHERE ADMLOGEXELOG\_ID= " + (Integer)globalMap.get("ADMLOGEXELOG\_ID") + ";*  *COMMIT;"* |

## Fin de traitement

Le « **TPOSTJOB »** déclenche une action requise après l'exécution d'un Job et on met à jour la log traitement, Etat = 0 (table ADMLOG\_EXECUTION\_TRAITEMENT) via la requête suivante :

|  |
| --- |
| "UPDATE admin.ADMLOG\_EXECUTION\_TRAITEMENT  SET ADMLOGEXETRT\_FIN\_DT = CURRENT\_TIMESTAMP, Etat = 0  WHERE ADMLOG\_EXECUTION\_TRAITEMEN\_id = " + globalMap.get("ADMLOG\_EXECUTION\_TRAITEMENT\_ID") + ";  COMMIT;" |

## Fin de la chaîne

Mise à jour du log d'exécution de chaîne courant, Etat = 1 (table ADMLOG\_EXECUTION\_CHAINE)

|  |
| --- |
| Etape0 Etape1 |

**Etape 0**

* Mise à jour la logue d’exécution

|  |
| --- |
| "UPDATE admin.ADMLOG\_EXECUTION\_CHAINE  SET ADMLOGEXECHN\_FIN\_DT = CURRENT\_TIMESTAMP,Etat = 1  WHERE ADMLOG\_EXECUTION\_CHAINE\_id = IDENT\_CURRENT('admin.ADMLOG\_EXECUTION\_CHAINE');  COMMIT;" |

**Etape1**

* Envoi d’un mail en cas d’erreur

# Traitements Staging

Le chargement de ces données provenant d’un fichier ou d’un groupe de fichiers se fait source par source de manière indépendante.

Les données sont stockées dans des tables STAGING brutes puis après, les différents contrôles, dans les STAGING finales.

## Phase 1 : STAGING brutes

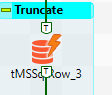
### STAGING brutes : Initialisation de la log

Le suivi du traitement d’extraction est initialisé dans les **tables :**

* **ADMLOG\_EXECUTION\_LOG**
* **ADMLOG\_ EXECUTION\_TRAITEMENT**

### STAGING brutes : Vidage

Avant le chargement dans les tables STAGING brutes, celles-ci sont tronquées via le composant « **TMSSQLROW »**,



Exemple :

|  |
| --- |
| TRUNCATE TABLE stabrute.collecte\_detail; |

### STAGING brutes : contrôle niveau 1

Au niveau de chaque fichier, on vérifie :

* La présence et la conformité de l’entête

|  |
| --- |
|  |

* Le nombre de champs attendus : il s’agit d’une vérification de ligne par ligne

|  |
| --- |
|  |

### STAGING brutes : Chargement

Chaque fichier est chargé dans la table associée et le nom du fichier se retrouve dans le nom de la table.

|  |
| --- |
|  |

Le suivi du traitement d’extraction est incrémenté dans la table **ADMLOG\_EXECUTION\_TRAITEMENT**

### STAGING brutes : contrôle niveau 2

Au niveau de chaque fichier, on vérifie le format des données conformément aux contrats d’interface :

* Format de date : DD/MM/YYYY HH24:MI:SS
* Séparateur des nombres décimaux : «.» (Point)
* La règle de présence obligatoire ou non de chaque champ de la table doit être respectée.
* Vérifier les types, la « NULLABILITÉ », la longueur de la ligne en fonction des valeurs de référence.

On utilisera le composant « **TSCHEMACOMPLIANCE »** pour valider toutes les lignes d'entrée en fonction du schéma de référence.

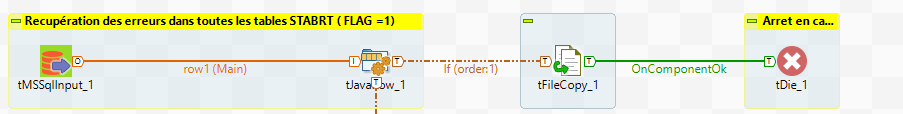
|  |
| --- |
|  |

Si une ligne ne respecte pas au moins une des règles de contrôle de niveau 2, son flag de contrôle vaut 1 sinon il vaut 0.

Si au moins une anomalie a été observée après le contrôle de tous les fichiers, les traitements s’arrêtent et **tous les fichiers du lot** seront archivés en REJET.

De la même façon, on chargera tous les autres fichiers en respectant tous les contrôles et règles.

NB : En cas d’erreur, celle-ci sera reportée dans la table des erreurs avec un suivi des logs conformément aux paragraphes dédiés.



### STAGING brutes : contrôle niveau 3

Au niveau de chaque fichier, on vérifie :

* Les contraintes d’intégrité référentielle
* Les règles métiers

Vérifier qu’un identifiant d’une table opérationnelle existe dans la table des références.

|  |
| --- |
|  |

Si une ligne ne respecte pas au moins une des règles de contrôle de niveau 3, son flag de contrôle vaut 1 sinon il vaut 0.

De la même façon, on chargera tous les autres fichiers en respectant tous les contrôles et règles.

NB : En cas d’erreur, celle-ci sera reportée dans la table des erreurs avec un suivi des logs conformément aux paragraphes dédiés.

### Contrôle des données des tables STAGING brut

On vérifie que, pour le lot considéré, aucune anomalie n’a été détectée sur aucun des fichiers.

Si au moins une anomalie a été observée sur au moins un fichier, les traitements s’arrêtent et les fichiers archivés en REJET.

|  |
| --- |
|  |

## Phase 2 : Chargement des tables STAGING final

Si aucune erreur n'est rencontrée lors des contrôles de niveau 1,2 et 3 sur le fichier, ce dernier est chargé en delta dans la table associée.

|  |  |
| --- | --- |
|  |  |

De la même façon, on chargera tous les autres fichiers en respectant tous les contrôles et règles.

NB : En cas d’erreur, celle-ci sera reportée dans la table des erreurs avec un suivi des logs conformément aux paragraphes dédiés

Mise à jour de la table des traitements :

A la fin du traitement des tables de STAGING finales, l’identifiant de la **table ADMLOG\_EXECUTION\_TRAITEMENT** passe de « 2 - En cours » à « 0 – succès » sinon « 1 –erreur ».

## Phase 4 : fin du job d’extraction des fichiers sources et archivage

A la fin de l’extraction du lot de fichiers, on archive ce dernier dans un répertoire d’archive (Voir le paragraphe 1.4)

Puis, en cas de bonne exécution, on fait un « **COMMIT** » sur l’ensemble des tables STAGING BRUTES + FINALES ?? sinon « **ROOLBACK** »

|  |
| --- |
| Etape0 Etape1 |

Etape 0 : composant de la fin de job

Etape1 : commit les tables STAGING

# Traitements ODS

L’objectif de cette étape est de faire l’amalgame dans un modèle métier unique des différentes sources de données hétérogènes :

* Définition d’un référentiel unique pour chaque notion : Client, Apporteur, Produit …
* Unification du format de stockage des données de ‘FAIT’.

Le modèle métier unique ne permet pas l’analyse, mais permet le stockage d’informations d’origine hétérogène dans un même modèle.

## Phase 1 : Démarrage du job de chargement de l’ODS

### Etape5 : Démarrage du job de chargement de ODS

APPLI. SOURCE: Wombat DB (ODS)

APPLI. CIBLE: Wombat DB (DataWarehouse)

### Etape6 : Chargement de ODS

Chargement des données des tables de STAGING finales dans les tables ODS.

Le chargement suivra un ordre bien précis.

Ainsi, on débutera par les tables de données de référence puis en respectant l’ordre correspondant au niveau de dépendance des tables.

### Etape6 : Mise à jour de la log

A la fin de chargement des tables ODS, on met à jour le champ état à 0 via la requête suivante :

|  |
| --- |
| UPDATE admin.ADMLOG\_EXECUTION\_TRAITEMENT  SET ADMLOGEXETRT\_FIN\_DT = CURRENT\_TIMESTAMP,  Etat = 0  WHERE ADMLOGEXETRT\_ID = " + (Integer)globalMap.get("ADMLOGEXETRT\_ID") + ";  COMMIT;" |

Puis, en cas de bonne exécution, on fait un « **COMMIT** » sur l’ensemble des tables ODS sinon « **ROOLBACK** »

# Traitements DWH

Le but est de mettre en forme les données du modèle métier pour les analyser.

Cela consiste à proposer différents indicateurs à analyser suivant différents axes d’analyse issus des référentiels construits à l’étape précédente

Le chargement de cette partie se fait en annule et remplace sur la période concernée par les informations modifiées dans l’ODS.

## Phase 1 : Démarrage du job de chargement de DWH.

SOURCE: Wombat DB (ODS)

APPLI. CIBLE: Wombat DB (DataWarehouse)

Chargement des données de l’ODS dans les tables de dimensions et les tables de faits

# Annexes usage des flags et commit

## Les flag

A travers tout le process d’alimentation des tables de STAGING, il est fait usage de « flag ».

Ces « flags » sont des informations enregistrés lors des contrôles de niveaux 2 et 3 et en corrélation avec le niveau de contrôle à savoir au niveau de la ligne.

Ainsi, si le format d’un champ n’est pas conforme (niveau 2) ou un champ ne respecte pas une contrainte d’intégrité référentielle, le « flag » de la ligne correspondante sera mis à jour.

Le « flag » permet donc de vérifier la conformité des données du fichier source et donc le pasage à l’étape suivante à savoir le chargement dans l’ODS.

Règles de Mise à jour

Commit