**Manuel d'utilisation**

**Description technique cadriciel Traitements Parallèle**

**Date : 28/09/15**

|  |  |
| --- | --- |
| **Identification du document** |  |
| Emetteur | Olivier Véhier |
| Référence |  |
| Date de création | 28/09/15 |
| Date de validation | 28/09/15 |

|  |  |
| --- | --- |
| **Responsable technique** | Olivier Véhier |

|  |  |  |
| --- | --- | --- |
| **Suivi des versions** | | |
| **Version** | **Date** | **Description** |
| 1 | 28/09/15 | Création du document |
| 2 | 09/07/2016 | Modification suite à upgrade 5.0 |

# Sommaire

[1. Lexique 4](#_Toc482625157)

[2. Spécification technique 5](#_Toc482625158)

[2.1. Besoin 5](#_Toc482625159)

[2.2. Objets techniques 5](#_Toc482625160)

[2.3. Classe de message dédié aux Tâches // - ZTASK 6](#_Toc482625161)

[2.4. Principe de fonctionnement 6](#_Toc482625162)

[2.5. Utilisation 6](#_Toc482625163)

[2.6. Description technique 9](#_Toc482625164)

[2.6.1. Classe Tâche - ZCL\_TASK 9](#_Toc482625165)

[2.6.1.1. Approche de conception 9](#_Toc482625166)

[2.6.1.2. Détails techniques 9](#_Toc482625167)

[2.6.1.2.1. Constructeur 9](#_Toc482625168)

[2.6.1.2.2. Lancement du traitement - TASK\_START 9](#_Toc482625169)

[2.6.1.2.3. Handler fin traitement - AT\_END\_OF\_TASK 10](#_Toc482625170)

[2.6.1.2.4. Conversion paramètres statiques en dynamiques - CONVERT\_STATIC\_PARAMETERS 10](#_Toc482625171)

[2.6.1.2.5. Conversion paramètres dynamiques en statiques- CONVERT\_DYNAMIC\_PARAMETERS 10](#_Toc482625172)

[2.6.2. Classe Manager de Tâche - ZCL\_TASK\_MANAGER 11](#_Toc482625173)

[2.6.2.1. Approche de conception 11](#_Toc482625174)

[2.6.2.2. Détails techniques 11](#_Toc482625175)

[2.6.2.2.1. Constructeur - CONSTRUCTOR 11](#_Toc482625176)

[2.6.2.2.2. Lancement nouvelle Tâche - TASK\_START\_NEW 11](#_Toc482625177)

[2.6.2.2.3. Attente fin de toutes les Tâches - TASK\_FINISH\_RUNNING 11](#_Toc482625178)

[2.6.2.2.4. Handler fin d'une Tâche - HANDLER\_END\_OF\_TASK 12](#_Toc482625179)

[2.6.3. Classe Handler Manager de Tâche 13](#_Toc482625180)

[2.6.3.1. Approche de conception 13](#_Toc482625181)

[2.6.3.2. Détails techniques 13](#_Toc482625182)

[2.6.3.2.1. Constructeur - CONSTRUCTOR 13](#_Toc482625183)

[2.6.3.2.2. Handler Fin d'une Tâche (Manager) - HANDLER\_END\_OF\_TASK 13](#_Toc482625184)

[2.6.3.2.3. Création Handler - HANDLER\_FACTORY 14](#_Toc482625185)

[2.7. Assistant d’utilisation 15](#_Toc482625186)

[2.8. Perfectibilité 16](#_Toc482625187)

[2.8.1. Problème connu 16](#_Toc482625188)

[2.8.2. Axe d'amélioration 16](#_Toc482625189)

[2.8.3. Reste à faire 16](#_Toc482625190)

[2.9. Exemple d'utilisation 16](#_Toc482625191)

1. Lexique

Dans ce document nous utiliserons certains mots, mais dont leurs définitions ne seront pas celles - nécessairement - usuellement utilisées.

Ainsi, voici quelques définitions des mots utilisées :

* **Tâche** : Processus en arrière-plan
* **Mode Dédié** : Lorsque le Manager de Tâche est initialisé avec un MF particulier
* **Traitement principal** : Fait référence au traitement utilisant le cadriciel.
* **Données de contextes (attachées)** : Données à transmettre au traitement de fin d'une Tâche.
* **Données statiques** : Données typées de façon formelle.
  + En effet, un MF RFC ne peut être appelé qu'avec des paramètres identifiés de façon formelle (Pas d'instance, pas de "TYPE REF TO")
* **Données dynamiques** : Données non typées de façon formelle (utilisation référence)

1. Spécification technique
   1. Besoin

Fournir au développeur un cadriciel pour paralléliser des traitements.

* 1. Objets techniques

Objets spécifiques utilisés :

|  |  |  |
| --- | --- | --- |
| Nom de l’objet | Type de l’objet | Commentaire |
| ZCL\_TASK\_MANAGER\_SIMPLIFY | Classe | Manager de Tâche –Version simplifié |
| ZCL\_TASK\_MANAGER | Classe | Manager de Tâche – Classe gestion Traitement // |
| ZCL\_TASK\_HANDLER | Classe | Tâche - Classe Handler |
| ZCL\_TASK\_CONTAINER | Classe | Tâche - Classe container de données |
| ZCL\_TASK | Classe | Tâche – Classe création traitement // |
| ZCL\_TASK\_TRACE | Classe | Tâche - Classe Trace |
| ZCL\_TASK\_PARAM | Classe | Tâche - Classe Exploitation Paramétrage traitement // |
| ZCL\_TYPE\_DEF\_AREAHANDLE\_ROOT | Classe SHM | Définitition Type – SM - Classe exploitation |
| ZCL\_TYPE\_DEF\_AREAHANDLE\_AREA | Classe SHM | Définitition Type – SHM - Classe exploitation |
| ZCL\_TYPE\_DEFINITION | Classe | Définitition Type - Classe utilitaire |
| ZCL\_TYPE\_DEFINITION\_CLUSTER | Classe | Définition de Type - Cluster - Classe utilitaire |
| ZCL\_TASK\_SHM\_AREA | Classe SHM | Tâche – SHM – Classe exploitation SHM – SHM Déf. Type |
| ZCL\_TASK\_SHM\_ROOT | Classe SHM | Tâche – SHM - Classe exploitation SHM Déf. Type |
| ZCL\_TASK\_SHM | Classe | Tâche – SHM - Classe utilitaire |
| ZTASK | Classe de Message | Tâche - Classe de Message |
| ZTEC\_TASK\_PARALLEL | GF | Tâche - Groupe-Fonction pour // |
| ZIF\_TASK\_CONSTANT | Interface | Tâche - Interface |
| ZCX\_TASK | Classe Exception | Tâche - Classe exception |
| ZCX\_TASK\_MANAGER | Classe Exception | Manager de Tâche - Classe Exception |
| ZCX\_TYPE\_DEFINITION | Classe Exception | Définition de Type - Classe Exception |
| ZCX\_TYPE\_DEFINITION\_CLUSTER | Classe Exception | Définition de Type – Cluster - Classe Exception |
| ZCLUST\_TYPE\_DEF | Table Transparante | Définition de Type - Cluster |
| ZTASK\_FUNC\_PARAM | Table Transparante | Tâche – Table de paramétrage |
| ZTASK\_TRACE\_H | Table Transparante | Tâche – Trace - Historique |

* 1. Classe de message dédié aux Tâches // - ZTASK

La classe de message "**ZTASK**" contient les messages destinés à ce cadriciel.

Nous utilisons des tranches de numéro afin de compartimenter la provenance :

* De **002** à **100** : **Tâche**
* De **102** à **200** : **Manager de Tâche**
* De **202** à **300** : **Handler de Tâche**
  1. Principe de fonctionnement



* 1. Utilisation

Voici le principe d'utilisation :

* La première étape (optionnelle) consiste à créer une routine ou une méthode qui sera appelée lorsqu'une Tâche sera **terminée**.
  + Attention, cette routine ou méthode doit **obligatoirement** respecter la signature suivante :
    - **Routine** :
      * USING
        + IS\_TASK\_CALL\_PROCESS type ZTEC\_S\_TASK\_CALL\_PROCESS
        + IT\_RESULT type ZIF\_TASK\_CONSTANT=>TT\_FUNCTION\_PARAMETERS
        + IT\_ATTACHED\_DATA type ZIF\_TASK\_CONSTANT=>TT\_TASK\_ATTACHED\_DATA
        + IV\_ERROR type XSDBOOLEAN
    - **Méthode** :
      * IMPORTING
        + IS\_TASK\_CALL\_PROCESS type ZTEC\_S\_TASK\_CALL\_PROCESS
        + IT\_RESULT type ZIF\_TASK\_CONSTANT=>TT\_FUNCTION\_PARAMETERS
        + IT\_ATTACHED\_DATA type ZIF\_TASK\_CONSTANT=>TT\_TASK\_ATTACHED\_DATA
        + IV\_ERROR type XSDBOOLEAN
* La seconde étape (optionnelle) consiste à créer une routine ou une méthode qui sera appelée **avant** qu’une Tâche soit lancée.
  + Attention, cette routine ou méthode doit **obligatoirement** respecter la signature suivante :
    - **Routine** :
      * USING
        + IS\_TASK\_CALL\_PROCESS type ZTEC\_S\_TASK\_CALL\_PROCESS
        + IO\_TASK\_CONTAINER type ref to ZCL\_TASK\_CONTAINER
    - **Méthode** :
      * IMPORTING
        + IS\_TASK\_CALL\_PROCESS type ZTEC\_S\_TASK\_CALL\_PROCESS
        + IO\_TASK\_CONTAINER type ref to ZCL\_TASK\_CONTAINER
* Création instance "**ZCL\_TASK\_MANAGER**" en précisant :
  + Groupe de Serveur
  + Nombre de Tâche maximales
  + Le nom du Module Fonction ou de la méthode Statique ou du Programme à appeler ou de la méthode statique - ce comportement est appelé "Mode Dédié"
    - Si renseigné à ce moment, il n'est pas nécessaire de la préciser lors du démarrage d'une nouvelle Tâche (voir explication correspondante)
  + Couple Routine / Programmer à appeler :
    - Avant le début d’une Tâche
    - Lors de la fin d'une Tâche
  + Couple Instance / Méthode à appeler :
    - Avant le début d’une Tâche
    - Lors de la fin d'une Tâche
  + L’activation des traces d’exécution
    - Mémoire ?
    - Trace ?
  + La compression des données
    - Ligne à ligne ?
    - Global ?
  + Gestion automatique de la SHM

Légende : Obligatoire, Facultatif, L'un ou l'autre ou aucun

* Constitution des données d'appel / de réception du MF à paralléliser
  + Appel méthode « TASK\_CONTAINER\_CREATE »
  + Utilisation méthode « ATTACHED\_DATA\_ADD »
  + Utilisation méthode « FUNCTION\_PARAMETER\_ADD »
  + Alimentation d'une table de type "**ZIF\_TASK\_CONSTANT=>TT\_FUNCTION\_PARAMETERS**" en remplissant les champs :
    - **PARAMETER\_NAME** : Avec le nom du paramètre (il peut s'agir d'un paramètre Import / Export / Changing /Table)
    - **PARAMETER\_VALUE** : Avec la valeur du paramètre (par référence !)
      * Cela ne sert à rien de remplir ce champ pour des paramètres d'EXPORT - de manière générale, il n'est pas utile de le renseigner, si c'est le MF à paralléliser qui l'alimente. **Sauf si la définition du paramètre d’appel est de type générique. Auquel cas vous pouvez alimenter soit cette zone, soit la suivante.**
    - **TYPE\_FOR\_NONE\_DDIC\_PARAMETER\*** : **À alimenter uniquement pour les paramètres dont le type est générique côté MF (ex : « ANY TABLE » ou « STANDARD TABLE ». Sauf si vous avez alimenté la zone précédente.**

**\*** : Dans le cas des paramètres génériques, nous recommandons l’alimentation de ce champ.

* Démarrage d'une nouvelle Tâche //
  + Appel de la méthode "**TASK\_START\_NEW**" en précisant :
    - Non mode dédiée :
      * Le nom du MF
      * Le nom de la Classe
      * Le nom de la Méthode
      * Le nom du Programme
    - La configuration de lancement de la Tâche
    - L’instance du Container
    - L’indicateur de surcharge de la configuration
      * Utile lorsque l’instance a été générée avec une certaine configuration et que l’on souhaite modifier ces paramètres pour une itération

Légende : Facultatif

* Si le traitement principal implique un traitement de fin d'une Tâche, il faut s'assurer que toutes les Tâches // soient terminées. Pour se faire, il suffit d'appeler la méthode "**TASK\_FINISH\_RUNNING**".
  1. Description technique
     + 1. Classe Manager de Tâche Simplifié - ZCL\_TASK\_MANAGER\_SIMPLIFY
       2. Approche de conception

Cette classe a été développée pour simplifier l’utilisation du Cadriciel de parallélisations lorsque le développeur souhaite l’implémenter sur un traitement unique (Mode dédié)

En effet, cette classe ne supporte l’appel qu’un seul et même traitement (méthode statique / module fonction / programme).

* + - 1. Détails techniques
         1. Exécution – EXECUTE

classe a été développée pour correspondre à un processus parallèle. Ainsi, elle se voit dotée de fonctionnalité portant autour de l'appel du processus

* + - * 1. Constructeur – CONSTRUCTOR

Création d’une instance du Manager de Tâche avec la configuration transmise par l’utilisateur.

* + - * 1. Lancement du traitement – PROCESS\_RUN

Parcours chaque entrée de la table de traitement pour l’ajouter en tant que paramètre d’appel au traitement dédié.

Appel de la tâche via la méthode « TASK\_START\_NEW » de l’instance du Manager de Tâche

* + - * 1. Traitement fin d’une Tâche – AT\_EOT

Ajout des données de retour du traitement dans l’attribut de l’instance

* + - * 1. Récupération des résultats – PROCESS\_RESULT\_GET

Cette classe a été développée pour correspondre à un processus parallèle. Ainsi, elle se voit dotée de fonctionnalité portant autour de l'appel du processus

* + 1. Classe Tâche - ZCL\_TASK
       1. Approche de conception

Cette classe a été développée pour correspondre à un processus parallèle. Ainsi, elle se voit dotée de fonctionnalité portant autour de l'appel du processus :

* Initialisation paramètres d'appel
  + Conversion des paramètres dynamiques en paramètres statiques (voir Lexique pour définition "dynamique" et "statique")
* Démarrage de la Tâche
* Événement fin traitement du processus
  + Conversion des paramètres statiques en paramètres dynamiques.

Par ailleurs, cette classe n'ayant pas d'intérêt "seule", il est possible de créer une instance uniquement dans la classe du Manager de Tâche.

* Génération d'instance définie sur "Privé"
* Le Manager de Tâche est défini comme Ami de la classe (afin de pouvoir accéder aux attributs et méthodes privés)
  + - 1. Détails techniques
         1. Constructeur
* Génération d'un ID unique de Tâche
* Initialisation attributs pour Mode Dédié
  + Initialisation attributs
  + Récupération paramètres d'appel du MF
    - * 1. Lancement du traitement - TASK\_START
* Initialisation indicateur Tâche en cours d'exécution
* Récupération paramètres d'appel du Traitement - Si non Mode Dédié
* Conversion des paramètres d'appel dynamiques en statiques
* Appel du MF "**Z\_TASK\_CALL\_PROCESS**" - dans un nouveau processus indépendant - en lui fournissant :
  + Le nom du MF **ou** (Classe / Méthode) à exécuter en //
  + Les paramètres d'appel
  + Les options liées à la compression

Légende : Obligatoire, Facultatif, L'un ou l'autre ou aucun

* En cas d'erreur
  + Relative au traitement asynchrone
    - Relance l'appel, mais dans pas un nouveau processus - appel synchrone / classique du MF
    - Conversion des paramètres statiques en dynamiques
    - Lève évènement fin d'une Tâche
  + Autre
    - Lève une Exception
      * 1. Handler fin traitement - AT\_END\_OF\_TASK

Cette méthode est appelée - par le système - lorsque le traitement asynchrone du MF "**Z\_TASK\_CALL\_PROCESS**" est terminé.

* Récupération des résultats
* Conversion des paramètres statiques en dynamiques
* Lève évènement fin d'une Tâche
  + - * 1. Conversion paramètres statiques en dynamiques - CONVERT\_STATIC\_PARAMETERS

Cette méthode sert à convertir les paramètres statiques en paramètre dynamique afin d'être exploité par le traitement principal.

* Pour chaque paramètre
  + Création de la donnée à partir de son type absolu
  + Suivant le genre de paramètre (table / structure / élémentaire / etc.)
    - Élémentaire :
      * Initialisation de sa valeur
    - Autres :
      * Décompression (CL\_ABAP\_ZIP)
      * Conversion XML -> Données
        1. Conversion paramètres dynamiques en statiques- CONVERT\_DYNAMIC\_PARAMETERS

Cette méthode sert à convertir les paramètres d'appel dynamiques en paramètre statique afin de pouvoir appeler le traitement en RFC.

* Pour chaque paramètre
  + Transcodification du type de paramètre (Import / Export / etc.)
    - Les valeurs pour le "type de paramètre" ne sont pas les mêmes que celles demandées pour l'appel dynamique de MF.
  + Récupération du type absolu des données associées
  + Dans le cas d’un paramètre de type générique
    - On stocke sa définition de type dans la mémoire partagée via l’utilisation de classe « **ZCL\_TYPE\_DEFINITION** ».
    - Si le développeur n’a pas renseigné le champ « **TYPE\_FOR\_NONE\_DDIC\_PARAMETER** », on émet un DUMP afin de mettre en évidence l’erreur d’implémentation.
  + Suivant le genre de paramètre (table / structure / élémentaire / etc.)
    - Élémentaire :
      * Initialisation de sa valeur statique
    - Autres :
      * Conversion Données -> XML
      * Compression du XML (CL\_ABAP\_ZIP)
    1. Classe Manager de Tâche - ZCL\_TASK\_MANAGER
       1. Approche de conception

Cette classe a été développée pour gérer l'ensemble des Tâches. Ainsi, elle ses méthodes se composent autour de la gestion des Tâches :

* Contrôle configuration pour la parallélisation
* Démarrage d'une Tâche
* Handler fin d'une Tâche
  + Transmet Évènement fin de traitement d'une Tâche

Par ailleurs, afin d'éviter que l'évènement de Fin d'une Tâche ne soit récupéré par une classe locale - pour éviter tout risque de mauvaise utilisation - l'évènement a été défini dans la partie privée de la classe. Par conséquent, la classe Handler est déclarée en Ami.

* + - 1. Détails techniques
         1. Constructeur - CONSTRUCTOR
* Contrôle :
  + Cohérence du Groupe de Serveur
  + Existence du MF - si Mode Dédié
* Création d'autant d'instances de Tâche que de nombres de Tâches maximales demandées (si non renseigné récupération nombre de tâche max sur le groupe de serveur indiqué).
  + Ajout de la Tâche dans une table avec l'ID en tant que Clef et avec "l'indicateur de Tâche en cours" en référence.
    - Le fait de passer par une référence nous permet d'avoir accès à la valeur réelle de l'indicateur. Ainsi, lorsque ce dernier est modifié, le Manager de Tâche le "sait" instantanément.
* Création de l'Handler de Tâche
  + - * 1. Lancement nouvelle Tâche - TASK\_START\_NEW

Cette méthode permet de lancer le traitement dans un nouveau processus.

* Récupération d'une Tâche libre
  + Si aucune disponible
    - Attente qu'une Tâche se libère
* Initialisation des données de contextes
* Trace l’appel (si activé)
* Lance le traitement en arrière-plan
  + Appel de la méthode "**TASK\_START**" de la Tâche.
    - * 1. Attente fin de toutes les Tâches - TASK\_FINISH\_RUNNING

Cette méthode sert à attendre la fin de toutes les Tâches encore en cours d'exécution.

* Tant qu'il reste des Tâches actives (indicateur) nous attendons.
  + - * 1. Handler fin d'une Tâche - HANDLER\_END\_OF\_TASK

Cette méthode est appelée - par le système - lorsque l'événement de fin d'une Tâche (au niveau de la classe des Tâches) est déclenché.

* Lève événement fin d'une Tâche - à destination de l'Handler.
* Libération mémoire des attributs de la Tâche
  + 1. Classe Handler Manager de Tâche
       1. Approche de conception

Cette classe a été développée afin de réceptionner l'évènement de fin d'une Tâche - provenant du Manager de Tâche. Elle permet d'exécuter le traitement spécifique à appliquer lorsqu'une Tâche se termine.

Tout comme la classe des Tâches, cette classe n'a pas d'intérêt "seule". Par conséquent, la génération d'instance a été définie sur "Privée". Cependant, afin d'éviter la redondance d'amitié entre l'Handler et le Manager, une méthode publique statique permet de générer une instance de l'Handler.

* + - 1. Détails techniques
         1. Constructeur - CONSTRUCTOR
* Initialisation des attributs
* Souscription à l'évènement de Fin d'une Tâche du Manager
  + - * 1. Handler Fin d'une Tâche (Manager) - HANDLER\_END\_OF\_TASK

Cette méthode est appelée - par le système - lorsque l'événement de fin d'une Tâche (au niveau de la classe Manager) est déclenché.

* Suivant le mode de traitement
  + Par routine :
    - Appel de la routine du programme spécifié avec les mêmes paramètres que l'évènement.
      * Nom du Module Fonction appelé
      * Données de retour
      * Données attachées
  + Par méthode :
    - Appel de la méthode de l'instance spécifiée avec les mêmes paramètres que l'évènement.
      * Nom du Module Fonction appelé
      * Données de retour
      * Données attachées
* En cas d'erreur on lève une exception qui ne doit pas être catchée afin de "forcer" un DUMP.
  + Nous justifions ce comportement par le fait que si une erreur survient à ce moment-là, c'est qu'il s'agit d'une mauvaise implémentation du cadriciel par le développeur. Par exemple : Mauvaise signature de la routine ou de la méthode.
    - * 1. Création Handler - HANDLER\_FACTORY

Cette méthode permet de générer une instance de l'Handler. Elle a été définie en tant que méthode statique publique afin d'éviter la redondance d'amitié entre l'Handler et le Manager.

* Création de l'instance de l'Handler (appel du constructeur)
  1. Assistant d’utilisation

Un assistant d’utilisation est disponible. Celui-ci permet aux développeurs de facilement intégrer le cadriciel à ces projets. (Supporte uniquement les « Programmes »)

* 1. Perfectibilité
     1. Problème connu
* Aucun problème connu, mais un souci potentiel identifié :
  + Forte volumétrie : **Indice de confiance 9,5/10**
    - Aucun problème avec plus de 600k lignes et 8 colonnes
    - Plus la volumétrie est importante plus le passage à la compression globale se fait pertinente. Dans la plupart des cas, la compression ligne / ligne suffit et est recommandée. Cependant, forcer la désactivation de la compression entraîne une perte de performance et accroît de manière significative la consommation mémoire.
* Surveiller la SHM
  + 1. Axe d'amélioration
* Aucun, au sens où nous ne savons pas ce qui est à améliorer et non pas que la cadriciel est parfait.
  + 1. Reste à faire
  1. Exemple d'utilisation

Le programme "**ZDEMO\_TASK**" sert d'exemple d'utilisation.