Noyau AMDA-NG - 2nde partie

DOSSIER DE CONCEPTION

Référence : CDPP-CD-32500-457-CS

Version 1.6 du 23/01/2014

afaq ISO 9001 Qualité AFNOR CERTIFICATION

CS SYSTEMES D'INFORMATION

22, avenue Galilée – 92350 le Plessis Robinson - Tél. : 01 41 28 40 00 – Fax : 01 41 28 40 40
www.c-s.fr

Tableau des signatures

	Nom	Fonction	Date	Signature
Préparé par :	N. Boursier		23/01/2014	
Validé par :	S.Frayssines		23/01/2014	
Approuvé par :	R.Patrier		23/01/2014	

Sommaire

Table des illustrations

Liste des figures

Liste des tableaux

Glossaire

Abréviation	Définition
AC-IS	Accord Cadre Informatique Spatiale
AMDA	Automated Multiple Dataset Analysis
AMDA-NG	Automated Multiiple Dataset Analysis - New Generation
CCTP	Cahier des Clauses Techniques Particulières
CDPP	Centre de Données de la Physique des Plasmas
CESR	Centre d'Études Spatiale des Rayonnements
CNES	Centre National d'Études Spatiales
CSSI	Communication et Systèmes – Systèmes d'Information.
IRAP	Institut de Recherche en Astrophysique et Planétologie
IHM	Interface Homme-Machine
NFS	Network File System
XML	eXtended Markup Language
STB	Spécification Technique de Besoins
US	User Story

1. INTRODUCTION

Ce document présente le travail de conception élaboré lors de la phase de conception globale puis lors des phases de conception détaillée de chacun des sprints du Noyau AMDA-NG – 2nde partie.

Ce document s'articule autour des grands chapitres suivants :

- ✓ Le présent chapitre décrit l'objet et la structure du document
- ✓ Le chapitre **4. Présentation de la conception du logiciel** présente le contexte AMDA ainsi que la méthodologie suivie lors de la conception.
- ✓ Le chapitre **5. Conception du logiciel** présente les travaux de conception proprement dits déclinés par composant et par module.

2. DOCUMENTS APPLICABLES

DA01	Consultation AC-IS N°DAJ/AR/EO-2013.08449 ‘‘Développement du noyau AMDA-NG - Seconde Partie’’. CDPP-CO-32500-452-CNES, 06/05/2013, Ed. 01 Rev 00.
DA02	Spécification de besoins techniques pour la seconde prestation du nouveau noyau AMDA. CDPP-ST-32500-451-CES, 13/05/2013, Ed. 01 Rev 00
DA03	Cahier des Clauses Techniques Particulières AC-IS. DCT/PS-2011-003173.
DA04	Exigence de réponse aux clauses de sécurité des Systèmes d'information de l'Accord Cadre AC-IS. DCT/PS-2010-15734
DA05	Projet de cahier des prescriptions de Sécurité des Systèmes d'Information Accord Cadre Informatique Spatiale Clauses Générique. DCT/PS-2011-003191
DA06	Exigences Normatives associées aux prestations de développement et de maintenance dans le domaine de l'informatique spatiales. ACIS-ACIBS-SP-GEN-1-CNES

3. DOCUMENTS DE REFERENCE

DR01	Dossier d'architecture du noyau d'AMDA-NG. CDPP-AR-32500-382-SI, Ed. 02 Rev. 01, 29/11/2012.
DR02	Dossier de conception du noyau AMDA-NG. CDPP-CD-32500-436-SI, Ed. 01 Rev. 06, 11/02/2013.
DR03	Dossier de contrôle des interfaces du noyau AMDA-NG. CDPP-IF-32500-438-SI, Ed. 01 Rev. 04, 05/02/2013.
DR04	Étude sur les solutions alternatives à IDL. CDPP-NT-32500-383-SI, Ed. 01 Rev. 02, 10/01/2010.
DR05	Manuel d'installation de AMDA Kernel. CDPP-MI-32500-440-SI, Ed. 01 Rev 05, 11/02/2013.

4. PRESENTATION DE LA CONCEPTION DU LOGICIEL

4.1Rappel du contexte

Développé de manière itérative depuis 2006, AMDA a les objectifs suivants :

- ✓ mettre en place une base contenant les données, étudiées dans le cadre de la physique des plasmas, issues de la collecte des satellites et des observatoires,
- ✓ proposer une interface web permettant l'accès à ces données,
- ✓ pour chaque utilisateur, centraliser l'ensemble de sa production dans un espace de travail accessible par le biais de l'interface web,

Pour ce faire, AMDA s'appuie sur l'existant issu du projet DD SYSTEM, notamment DD client et DD server

La première partie du développement du noyau AMDA-NG a permis de définir les bases de l'architecture de l'application. Elle s'est concentrée sur les aspects accès aux données (notamment depuis DD Server) et sur la mise en place d'un framework permettant de faciliter l'ajout de fonctionnalités.

L'objectif de cette nouvelle prestation est de compléter cette première instance avec les fonctionnalités suivantes :

- ✓ gestion des données sous forme de « Time Table » (tables d'évènements),
- ✓ ajout d'une requête conditionnelle « Data Mining » produisant des « Time Table »,
- ✓ ajout de post-traitements éventuels sur les résultats d'une requête,
- ✓ complétion du module existant « download »,
  - ➢ téléchargement d'une « Time Table »
  - ➢ activation d'un post traitement
- ✓ mise en place d'un module « Plot » produisant des tracés de données.

4.2Architecture statique

Ce paragraphe décrit le fonctionnement global du système AMDA. Le schéma ci-dessous présente l'architecture générale du système AMDA :

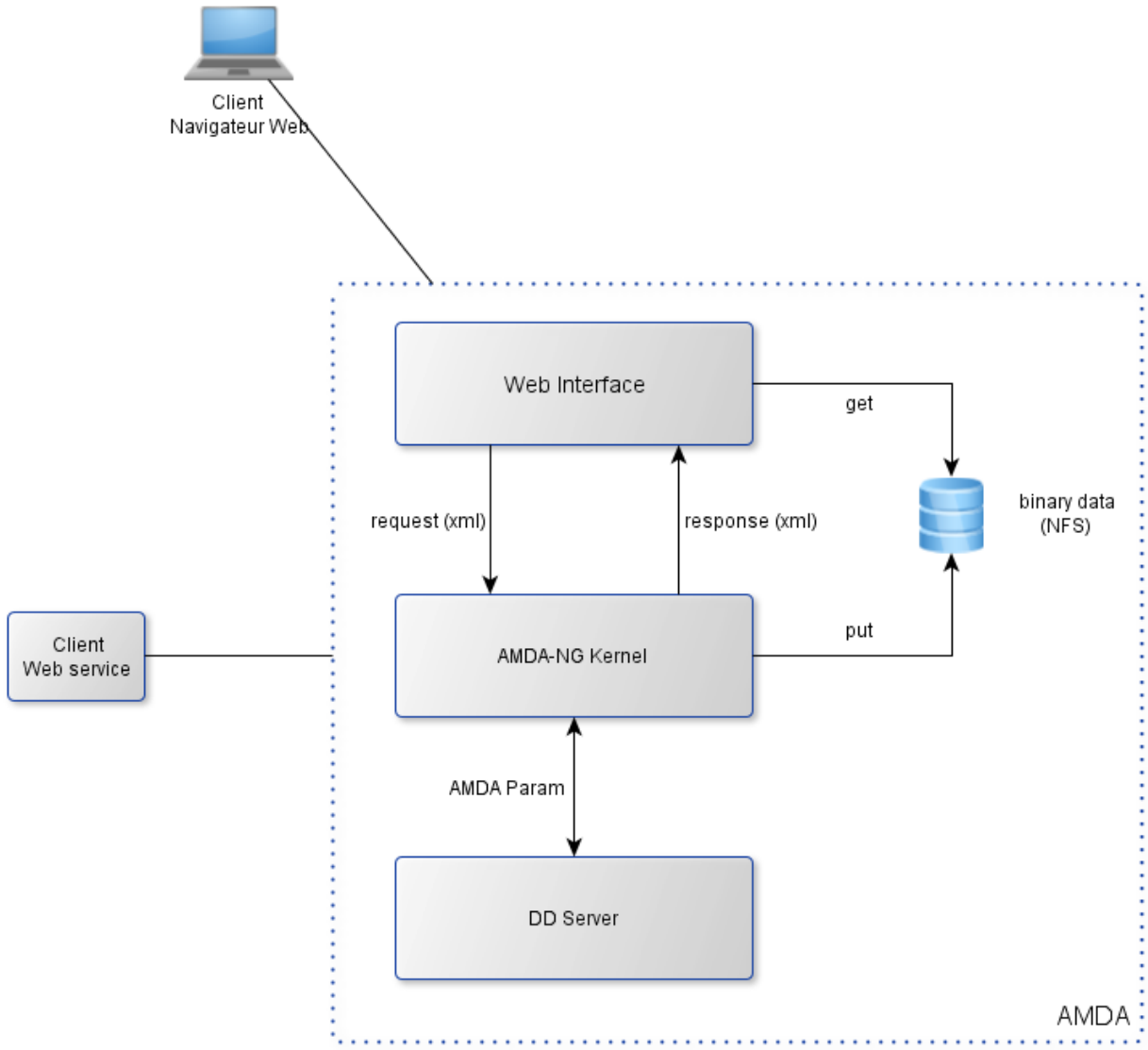


Figure 1 : Architecture générale du système AMDA

1. Un utilisateur saisit une demande via un client IHM web (EXT-JS).
2. Cette demande est traitée par l'interface « Web Interface » qui la transmet sous la forme d'une requête XML au noyau AMDA (AMDA-NG Kernel).
3. Ce dernier traite la requête …
4. … en s'appuyant sur le serveur DD Server pour obtenir les données relatives aux valeurs des paramètres AMDA …
5. … puis retourne une réponse, au format XML également, à l'interface « Web Interface ».
6. Finalement, l'interface « Web Interface » émet la réponse à destination du poste client.
7. Lorsque les données retournées sont sous forme binaire (plot ou zip par exemple), celles-ci sont stockées par le noyau dans un espace partagé (NFS) puis récupérées (consommées) par l'interface Web.

4.3 Contexte des interfaces

Les interfaces externes identifiées pour le Noyau AMDA-NG sont la « Web Interface », le « DD Server » et l'espace NFS. Ces interfaces sont détaillées dans le Dossier de Contrôle des Interfaces [IF].

4.4 Démarche globale de conception

La conception préliminaire du système s'attache à décrire "comment les nouvelles fonctionnalités vont s'intégrer au noyau existant". Elle permet d'aborder de façon macroscopique toutes les grandes fonctions du système au travers des scénarios afin de construire la dynamique entre les éléments constitutifs du système et ainsi identifier les besoins en termes d'interfaces pour chacun d'eux.

Les points de départ de la conception sont la spécification de besoin technique [DA02] et, au fur et à mesure de l'avancement du projet, la description des « stories » (scénarios) élaborées dans le cadre de la démarche Agile adoptée pour sa réalisation.

4.4.1 Concepts UML utilisés

Des diagrammes UML sont utilisés tout au long du document pour exposer les éléments de conception. Le tableau ci-dessous présente la définition des concepts utilisés :

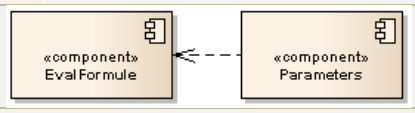
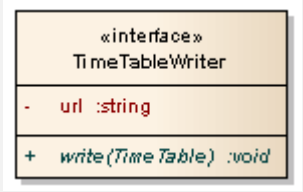
Notion	Définition	Représentation/Exemple
Composant	Ce stéréotype désigne un composant du système. Un composant peut être lui-même composé de plusieurs autres composants.	
Dépendance	Ce lien désigne une dépendance d'un composant vers un autre composant.	
Diagramme de composants	Diagramme représentant les dépendances entres composants.	
Classe	Une classe est un ensemble d'attributs représentant l'état d'un objet et de méthodes représentant ses comportements. Elle peut être caractérisée par des stéréotypes (ici « interface »)	
Implémentation d'interface	Ce lien indique que le composant (ou module) réalise une implémentation de l'interface reliée.	
Agrégation	Une agrégation est une association qui exprime une relation de type ensemble/élément. Elle n'implique pas de notion d'appartenance.	
Composition	Une composition est une agrégation forte : la destruction de l'agrégat implique la destruction des objets agrégés.	

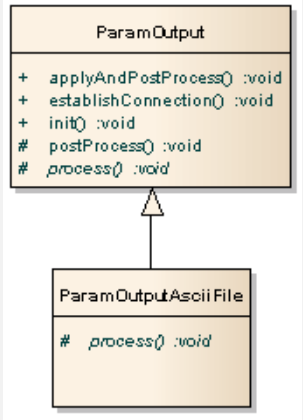
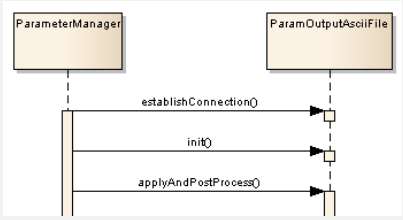
Diagramme de classes	Diagramme présentant les différentes classes et interface du système ainsi que leurs relations.	
Diagramme de séquence	Diagramme représentant les interactions entre le système et les acteurs en montrant les messages échangés et la chronologie des enchaînements. Le diagramme de séquence est utilisé pour illustrer un scénario mettant en œuvre les classes d'un paquetage et la dynamique entre les composants.	

Tableau 1 : Concepts UML

4.4.2 Règles de nommage

D'une manière générale, les fichiers sources C++ ont pour extension `.cc` et les fichiers header `.hh`.

D'autre part, les objets spécifiques respectent les conventions de nommage suivantes (ces conventions sont déduites de l'existant et sont limitées au périmètre de la prestation courante) :

- ✓ Les implémentations de `ParamOutput` sont préfixées par `ParamOutput`,
- ✓ Les implémentations de `AMDA::XMLConfigurator::NodeGrpCfg` sont post-fixées de `Node`.

4.4.3 Traçabilité

Les exigences de spécifications techniques complétées des User Stories sont rappelées pour chacun des composants développés dans le cadre de la prestation. Un tableau de la forme suivante est présenté au début du chapitre de conception de chacun des composants :

Id	Description	Commentaire
AMDA_<id_composant>_<num>	✓ Texte de l'exigence ou du besoin	*Origine du besoin (STB, US)*

Tableau 2 : Exemple de rappel des exigences

5. CONCEPTION DU LOGICIEL

5.1Généralités

5.2Architecture logicielle

Le diagramme suivant présente les différents composants du noyau AMDA-NG répartis en couche :

- **Couche service**
  - Rôle : expose des services à la couche Application (ici matérialisée par la « Web Interface) et convertit les données métier en données affichables
- **Couche Métier (Business)**
  - Rôle : met en œuvre de la logique applicative
- **Couche Persistance (Persistence)**
  - Rôle : permet d'accès aux données persistantes

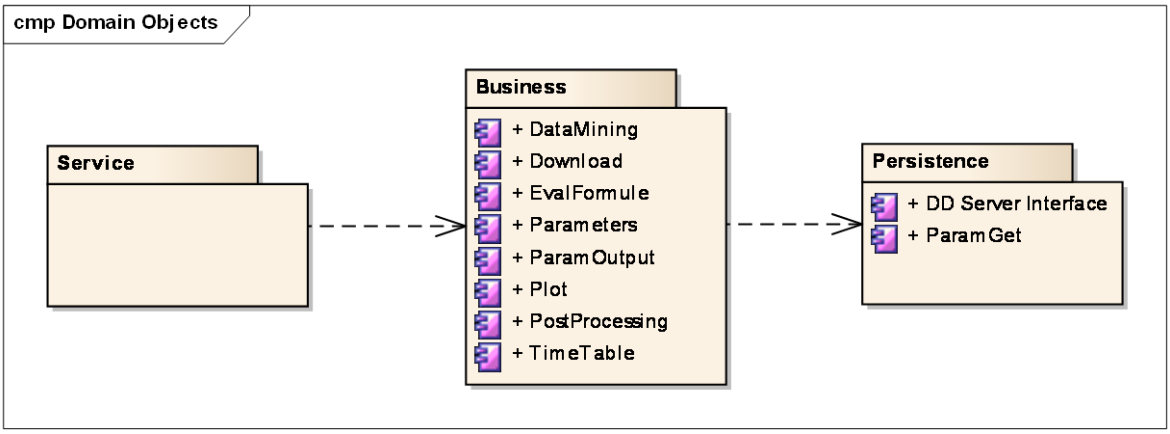


Figure 2 : Architecture logicielle de noyau AMDA-NG

Le noyau AMDA-NG se présente sous la forme d'un exécutable, qui s'appuie sur des librairies dynamiques, permettant de traiter une requête : lorsque le noyau « parse » la requête, il crée dynamiquement les objets nécessaires à son traitement.

Le noyau AMDA-NG propose un mécanisme de **« plugin »** permettant de prendre en compte de nouvelles requêtes de manière dynamique : il n'est pas nécessaire d'arrêter le serveur pour charger un nouveau « plugin ». Ces « plugins » rendent le noyau AMDA-NG extensible. Ils portent sur :

- l'**accès aux données** (`ParamGet`) via DD Server,
- la **production de résultats** (`ParamOutput`) ; ce type d'extension est essentiel car il permet de traiter les différents types de requêtes,
- le **traitement des données** (`Process`) ; cette extension permet de mettre en place des fonctions supplémentaires sur les paramètres avant de produire les résultats (par exemple calcul d'expressions mathématiques).

La mise en place des plugins s'appuie sur un framework dédié. Techniquement, les plugins sont implémentés sous forme de librairie dynamique (.so) permettant une édition de lien à l'exécution entre le code appelant et les fonctions du plugin.

Au niveau des requêtes et de la définition des paramètres AMDA, le mécanisme d'extension est pris en compte par l'utilisation de sous-schémas XSD : un schéma spécifique est défini pour chaque type

d'extension concret et intégré au schéma global de la requête via le mécanisme de « substitution group ». Ces schémas et sous-schémas sont décrits dans le Dossier de Contrôle des Interfaces [IF].

Les développements nécessaires pour réaliser les nouvelles fonctionnalités s'appuient largement sur cette architecture pour s'intégrer au mieux dans le système actuel.
Ainsi :

- ✓ Les composants **DataMining** et **Plot** sont développés comme de nouveaux plugins de type « ParamOutput »).
- ✓ Le composant **PostProcessing** est développé comme un nouveau plugin sur le même principe que les modules existants.
- ✓ Le module existant **Download** est modifié pour prendre en compte les nouveaux besoins.
- ✓ Seule la prise en compte des tables d'évènements (composant **TimeTable**) consiste en une modification du cœur du noyau AMDA-NG.

5.3Conception des composants logiciel

5.3.1 Composants existants

Ci-après, une brève description des divers composants déjà existants.

Couche Métier (Business)

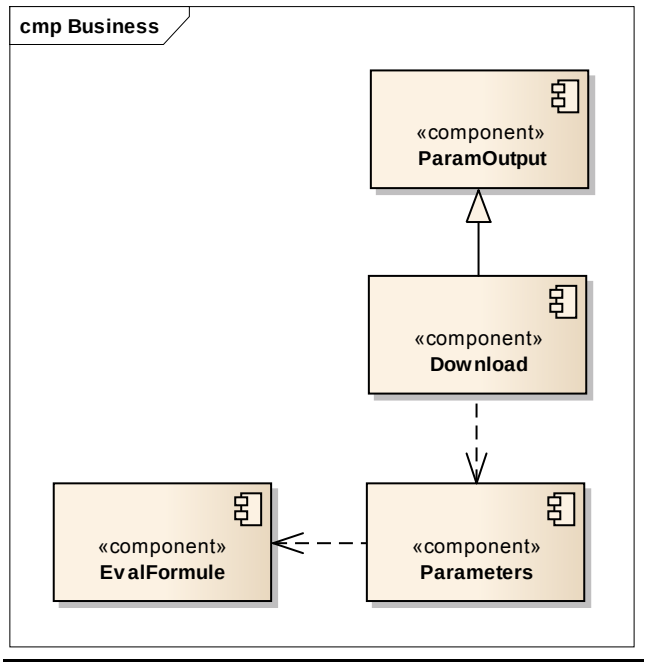


Figure 3 : Composants de la couche Métier

- ✓ ParamOutput
  - ➢ Interface de définition des divers types de sortie
- ✓ Download
  - ➢ Permet de mettre à disposition des fichiers de sortie au format ASCII
- ✓ Parameters
  - ➢ Responsable de l'accès en lecture et en écriture aux paramètres de la base (via DDClient/DDServer)
  - ➢ Permet la construction des paramètres composés

- Autorise les calculs sur les paramètres

✓ EvalFormule
- Valide et évalue les formules de la balise `process` (s'appuie sur les bibliothèques Boost)

Couche Persistance

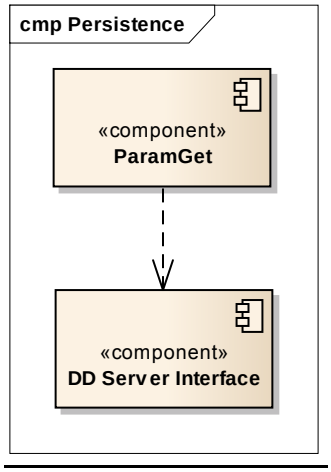


Figure 4 : Composants de la couche Persistance

✓ DD server Interface
- Gère l'accès à DD Server via des sockets TCP/IP

✓ ParamGet
- Pilote l'accès aux données via DD Server Interface

Composant transverse

✓ Trace :
- Propose un mécanisme de log basé sur **log4cxx** d'Apache

5.3.2 Intervalle de temps de recherche

Actuellement le système récupère les données de un ou plusieurs paramètres sur un unique intervalle de recherche. Celui-ci doit maintenant être capable de prendre en compte une liste d'intervalle de recherche sur lesquels récupérer les données de un ou plusieurs paramètres.

La mise en œuvre de ce procédé nécessite dans un premier temps de modifier le schéma « interval.xsd » pour prendre en compte le type suivant de structure XML :

```
<timetable id="tt_1.txt"/>
```

Il s'agit d'une unique balise qui identifie le nom du fichier TimeTable (cf. **5.3.4**) sur lequel doit être récupéré tous les intervalles de temps qui le constituent.

L'identifiant de la TimeTable référencée dans cette balise peut être un simple nom de fichier, un chemin vers ce fichier ou encore une URL http définissant sur quel serveur se trouve le fichier.

Et maintenant la voici dans son contexte :

```
<request>
	<params>
		<param id="velocity_cond"/>
	</params>
	<times>
		<timetable id="tt_1.txt"/>
	</times>
	<outputs>
		<dataMining>
			<fileFormat>XML</fileFormat>
			<timeFormat>ISO</timeFormat>
			<outputStructure>one-file</outputStructure>
		</dataMining>
	</outputs>
</request>
```

La lecture d'un tel fichier se fait au niveau du composant `XMLRequest` (plus précisément la classe `XMLRequestParser`) dans laquelle doit être définie une classe `TimeTableNode` (comme cela est fait pour la balise `startTime` et `timeInterval`). Une fois les intervalles de temps récupérés ceux-ci sont stockés par la classe `ParameterManager` qui est en charge d'initialiser les classes dérivées de `ParamOutput` (tels que le composant DataMining ou Download).

5.3.2.1 Initialisation des paramètres

La généralisation à une liste d'intervalles de temps nécessite la modification de toutes les classes (attributs et signatures des fonctions `init`) participant à la phase d'initialisation d'une requête :

- ✓ `Parameter` et `ParamInterval`,
- ✓ `DataWriter`, `ProcessStandard`, `ProcessSamplingClassic`, `ParamGetDDBase` et les librairies externes,
- ✓ `VirtualInstrument`, `VirtualInstrumentInterval`.

Ces modifications permettent de descendre jusqu'à la couche la plus basse (intéraction directe avec le serveur) la liste des intervalles de temps qui sera gérée par la classe `VirtualInstrumentInterval`.

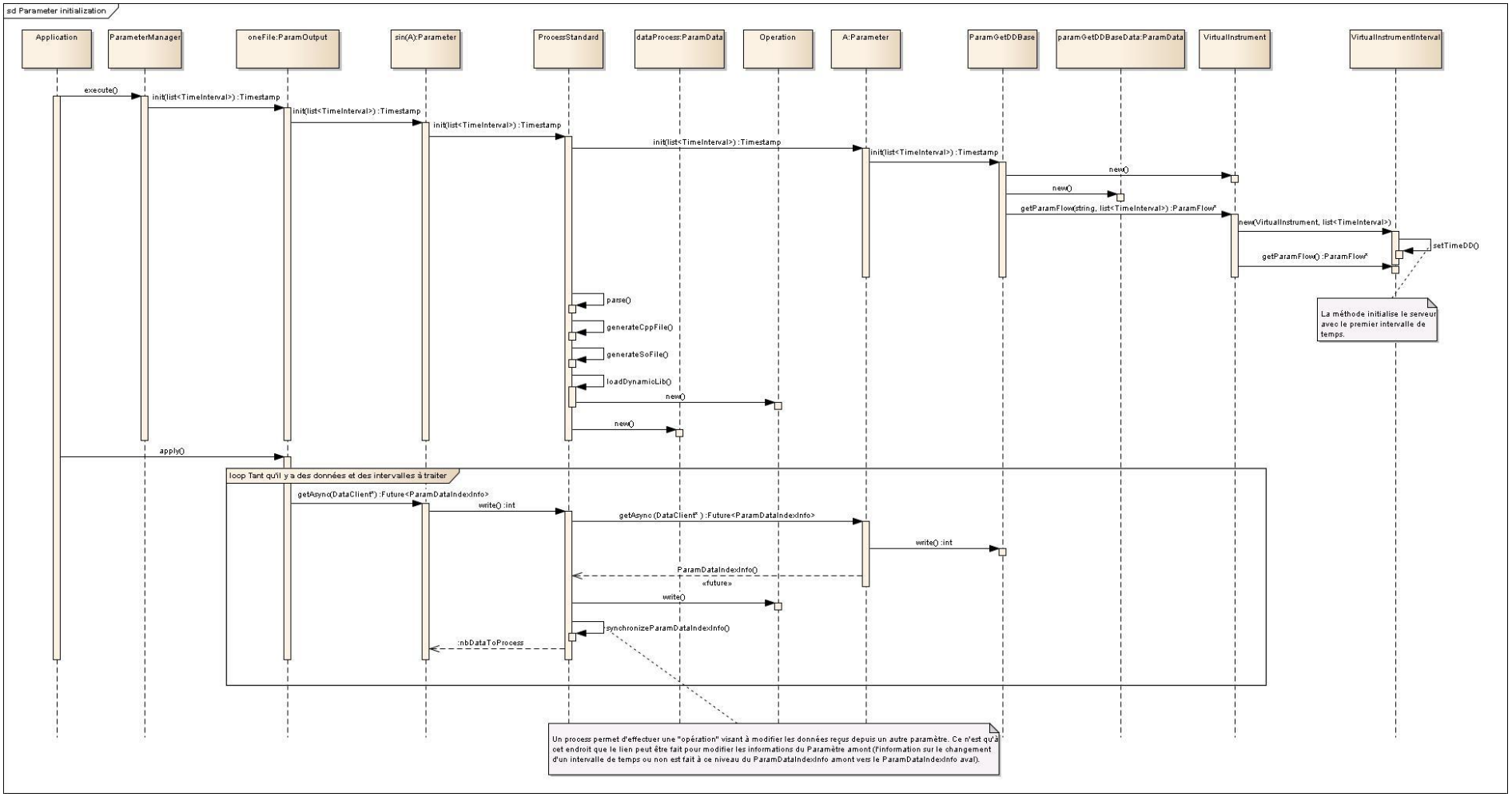


Figure 5 : Diagramme de séquence de l'initialisation d'une requête

Le mécanisme de récupération des données reste sensiblement le même, tant que l'instance de `ParamOutput` reçoit des données celle-ci appelle la méthode `getAsync()` qui envoie une nouvelle requête au serveur pour récupérer la suite des données et ce jusqu'à ce qu'il y ait famine.

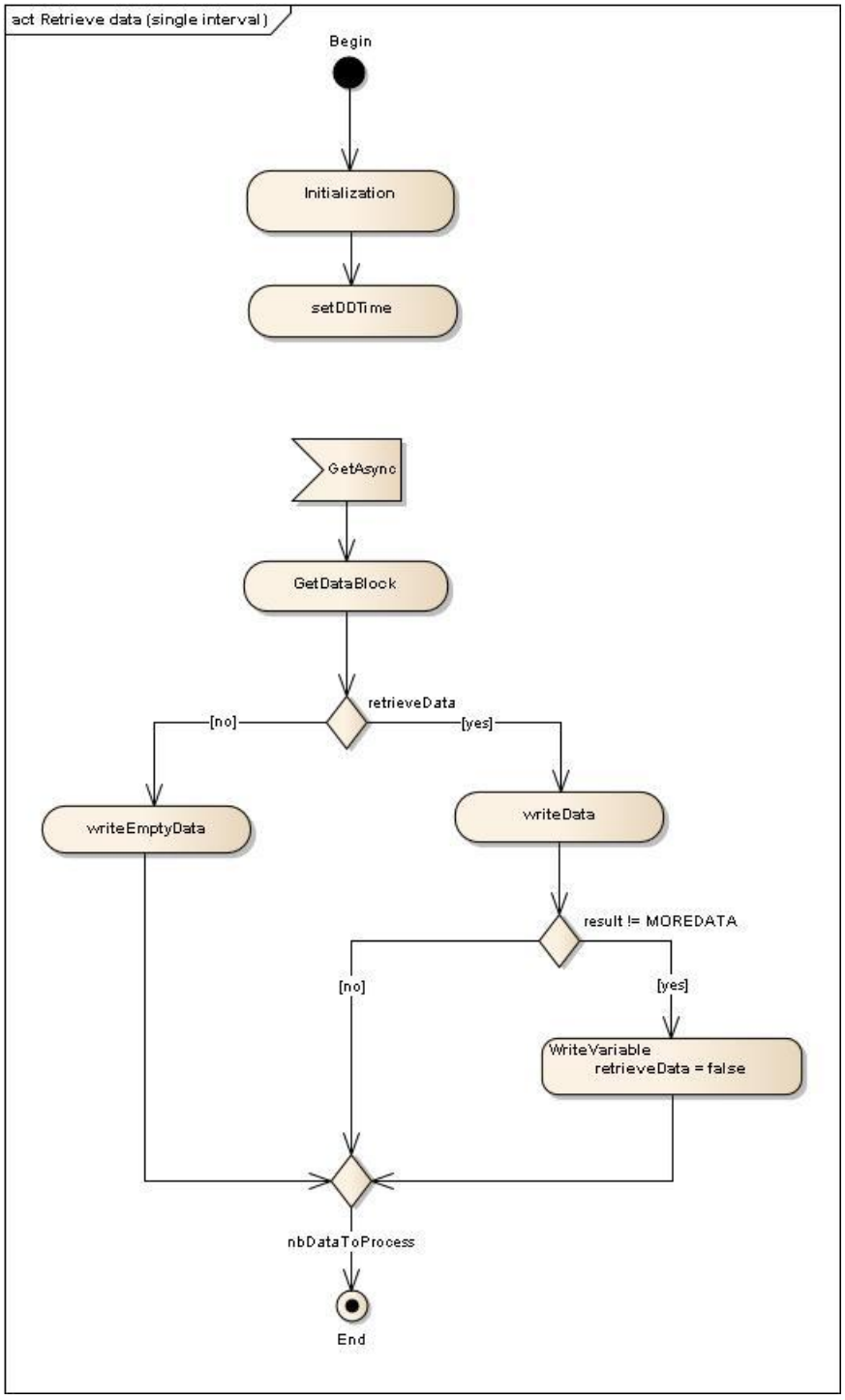


Figure 6 : Mécanisme de récupération de données sur un seul intervalle

La prise en compte de plusieurs intervalles de temps a peu d'impact sur le mécanisme de départ. Lorsqu'il n'y a plus de données à écrire (section `writeEmptyData`), une vérification sera faite pour savoir si tous les intervalles de temps ont été traités. Si ce n'est pas le cas, le serveur est réinitialisé (changement de la date de départ et de la durée) puis la classe `VirtualInstrumentInerval` se met en attente pour récupérer les données sur le prochain intervalle de temps.

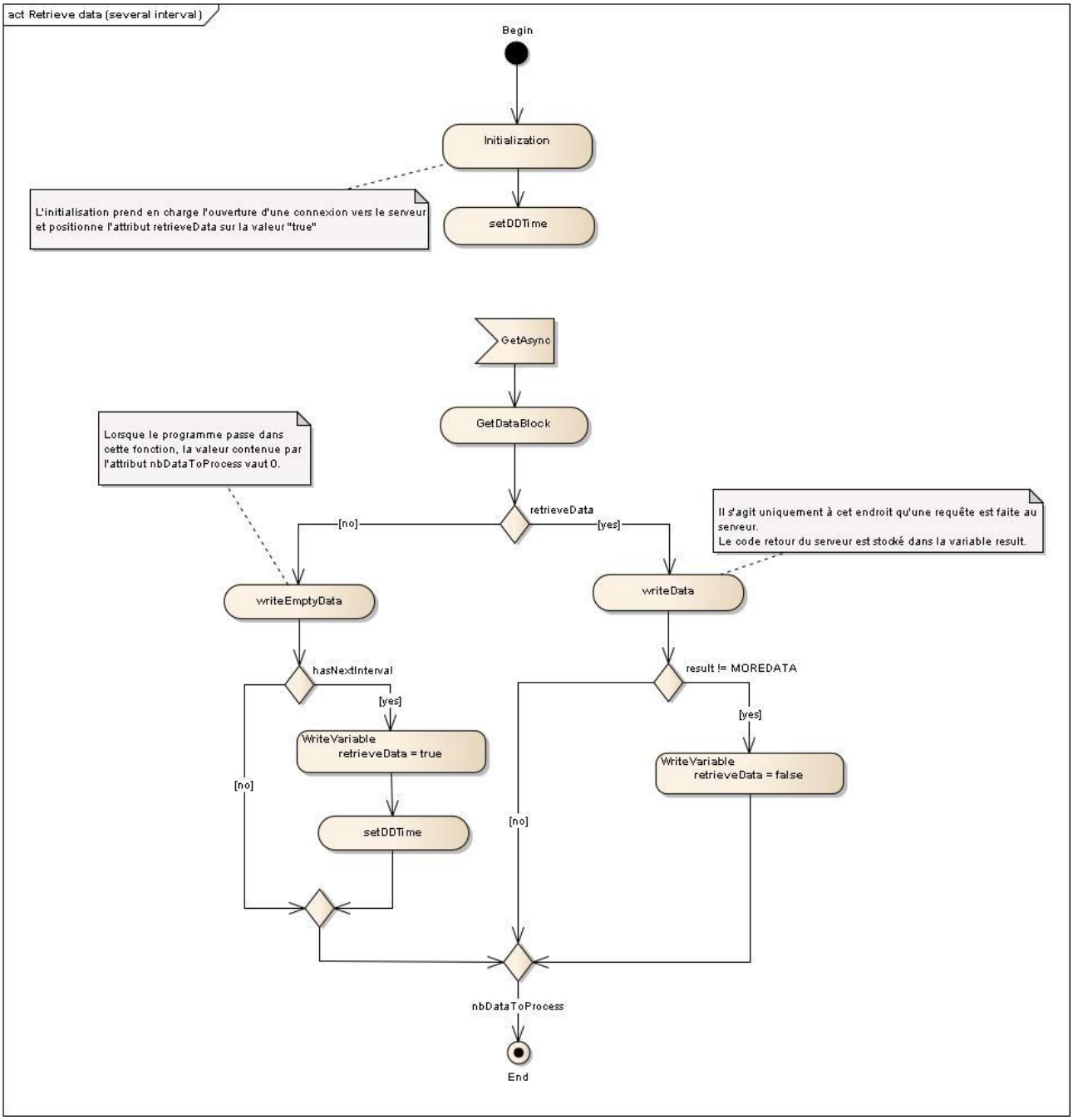


Figure 7 : Mécanisme de récupération des données sur plusieurs intervalles

Avec ce type de solution la classe `VirtualInstrumentInterval` renvoie toujours deux états distincts :

- ✓ Il n'y a plus de données sur cet intervalle de temps mais il reste des intervalles de temps à traiter,
- ✓ Il n'y a plus de données et tous les intervalles de temps ont été traités.

Pour que l'instance de haut niveau (ParamOutput) soit au courant du changement d'intervalle de temps, cela nécessite d'effectuer une « synchronisation » des données.

5.3.2.2 Récupération des données et synchronisation

La remontée de l'information propre au changement d'intervalle de temps se fait par le biais de la classe `ParamDataIndexInfo` qui doit être « synchronisée » entre les différentes couches de paramètres.

La récupération de données consiste en un chaînage de paramètres sur lesquels est associé un process capable de transformer les données reçues en entrée. Sur chaque paramètre, il existe différentes instances de `ParamDataIndexInfo` :

- ✓ une instance liée au paramètre amont et qui contient donc l'information de changement d'intervalles de temps,
- ✓ une instance liée au paramètre actuel,
- ✓ une instance liée au paramètre ou client aval désireux de récupérer les données du paramètres courant.

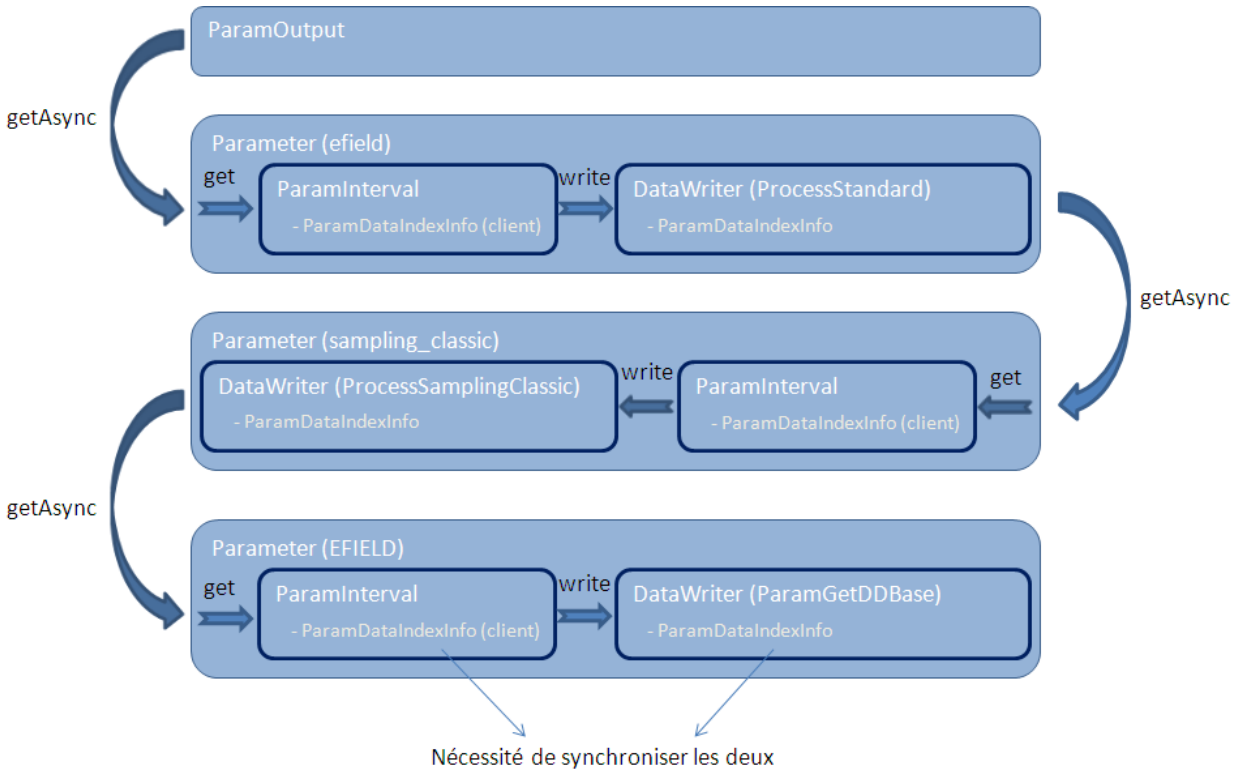


Figure 8 : Chaînage de paramètres

Pour que l'information remonte jusqu'à l'instance de `ParamOutput`, deux niveaux de synchronisation sont nécessaires :

- ✓ le premier concerne l'instance contenue dans `ParamInterval` et celui obtenu à la sortie du process (`DataWriter`) qui est une synchronisation interne (à l'intérieur du paramètre),
- ✓ le second concerne l'instance reçu du paramètre aval (par exemple pour le paramètre efield il s'agit de l'instance du paramètre sampling_classic) qui est une synchronisation externe (entre deux paramètres).

De cette manière, la synchronisation du premier cas est effectuée dans la méthode `get()` de la classe `ParamInterval` pour synchroniser la classe `ParamDataIndexInfo` du client avec celle de la classe `DataWriter` après avoir appelé la méthode `write()` sur celui-ci.
Pour le second cas, le synchronisation se passe dans la méthode `write()` de la classe `DataWriter` qui doit se faire uniquement après avoir effectué une opération de transformation (ré-échantillonnage, …) des données reçues par la paramètres amont.

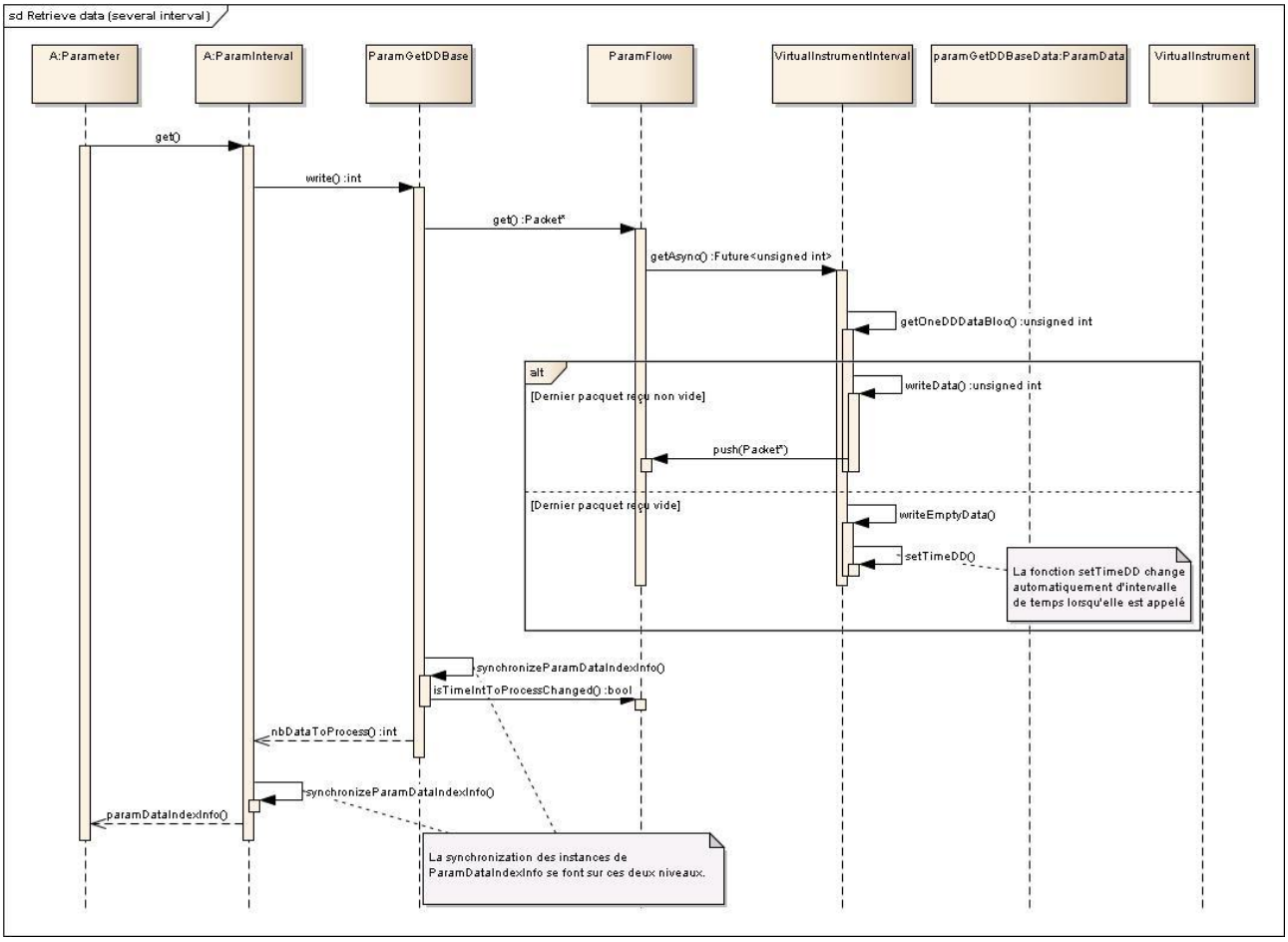


Figure 9 : Diagramme de séquence de récupération des données sur un paramètre

Au niveau de l'instance de ParamOutput si celui-ci lit une information de changement d'intervalle de temps, cela veut dire que le dernier pacquet reçu doit être traité dans l'intervalle de temps courant (quelque soient les données si c'est vide ou non vide) et doit passer à l'intervalle de temps suivant.
A contrario, s'il n'y a plus de données et que l'intervalle de temps n'a pas changé, cela veut dire que tout les intervalles de temps ont été traités.

5.3.3 Conception générale des nouveaux plugins

Dans le cadre des nouveaux développements, des composants de types plugins sont réalisés. Ces plugins sont des librairies dynamiques (.so) placées dans un répertoire dédié connu de l'application (propriété `app.plugin` du fichier de configuration `app.properties`.

Chaque plugin contient une classe `AMDAPlugin` qui présente deux méthodes attendues pour son chargement automatique :

- ✓ `registerPlugin` (enregistrement des implémentations pour les nouvelles fonctionnalités)
- ✓ `getPluginVersion` (récupération de la version de plugin)

5.3.4 Composant « Time Table »

5.3.4.1 Exigences techniques

Id	Description	Commentaire
AMDA_TT_010	Le système doit pouvoir lire et écrire des time tables aux formats suivants : ✓ ASCII ✓ XML (VOTable 1.3 - http://www.ivoa.net/documents/VOTable)	STB §3, US2
AMDA_TT_011	Le système doit pouvoir lire et écrire une time table au format XML interne AMDA	US2
AMDA_TT_012	Le système doit pouvoir lire une time table depuis une URL de la forme http://path/to/tt	US43
AMDA_TT_020	Une time table est définie par une liste d'intervalles, une description, une date de création, un nom et un historique.	STB §3, US2
AMDA_TT_030	Une date d'intervalle est exprimée au format ISO : YYYY-MM-DDTHH:MM:SS[.MSK]	STB §3, US2
AMDA_TT_040	Le système offre les opérations suivantes entre 2 time tables : fusion, intersection	US3
AMDA_TT_042	Le système offre une opération complémentaire à l'opération d'intersection.	US3
AMDA_TT_043	Le système propose des exécutables standalone permettant d'appeler les opérations de fusion, d'intersection et leurs inverses	US3

Tableau 3 : Exigences du composant Time Table

5.3.4.2 Objet

Le composant Time Table est responsable de la lecture et de l'écriture de fichiers Time Table. Il gère les formats suivants :

- ✓ ASCII
- ✓ VOTable (XML)
- ✓ Format XML interne AMDA

Dans ces trois formats, la date est exprimée en utilisant le format ISO YYYY-MM-DDTHH:MM:SS[.MSK].
Exemples : 2008-02-26T15:00:23, 2008-02-26T15:00:23.123.

5.3.4.3 Classe TimeTable

Une `TimeTable` est définie par :

- ✓ un nom,
- ✓ une date de création,
- ✓ une description,
- ✓ un historique,
- ✓ et une liste d'intervalles

Il est possible d'effectuer des opérations d'union et d'intersection ainsi que sa complémentaire entre plusieurs `TimeTable`.

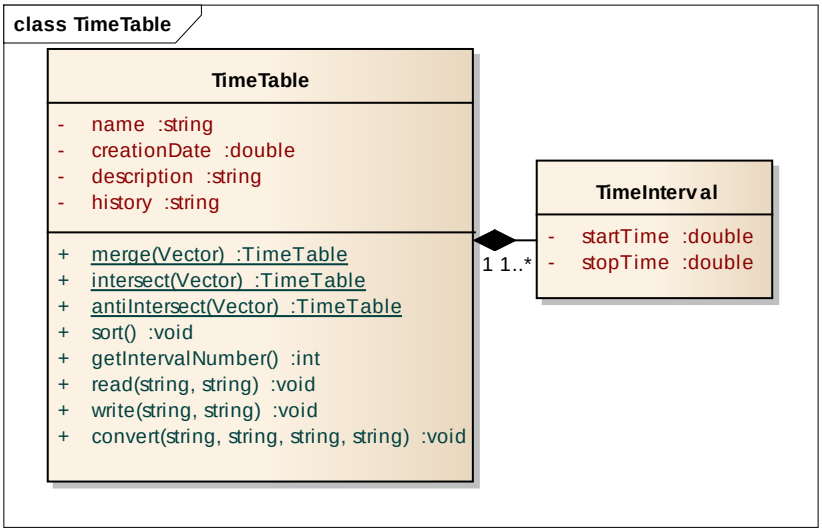


Figure 10 : Définition d'une TimeTable

5.3.4.4 Opérations sur les TimeTable

Il est possible d'effectuer les operations suivantes :

- ✓ Union (`merge`) sur 1 … N time tables
- ✓ Intersection (`intersect`) sur 2 … N time tables
- ✓ Inverse de l'intersection (`antiIntersect`) sur 2 … N time tables

La figure suivante schématise le fonctionnement des opérations de fusion, d'intersection et de sa complémentaire. Chaque bande horizontale noire symbolise un intervalle, le cadre en pointillé symbolise le ou les intervalles résultant de l'union, de l'intersection ou de l'antiintersection de deux intervalles.

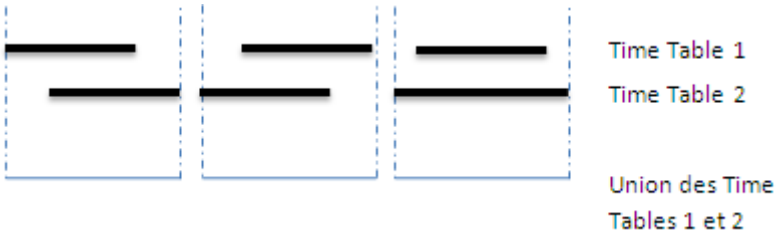


Figure 11 : Opération Union

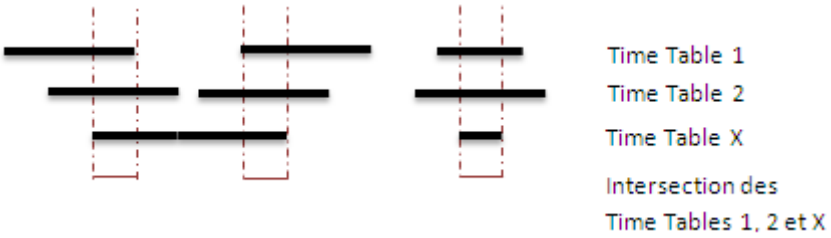


Figure 12 : Opération Intersection

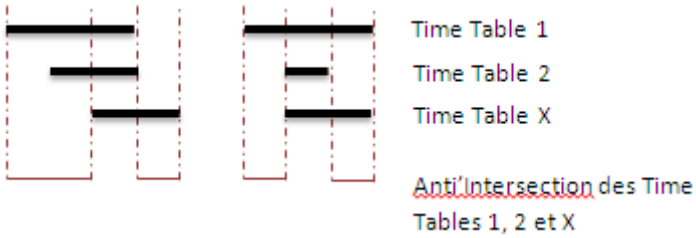


Figure 13 : Opération Anti'Intersection

La timetable ainsi créée présente les métadonnées suivantes :

- ✓ **Nom :** "tt1_i_tt2_i_tt3 … "/ "tt1_u_tt2_u_tt3…"/"tt1_ai_tt2_ai_tt3 … "
- ✓ **Description :** "Intersection between …" / "Union between …"/"Anti'intersection between …"
- ✓ Historique : vide
- ✓ Date de creation : date courante

Par ailleurs, ces operations sont accessibles via un executable `ttOperations` qui s'utilise comme suit :

```
ttOperations OPERATION [OPTIONS] TT...
avec :
```

- OPERATION parmi (intersect, union, antiintersect)
- OPTIONS
  - --with-msk pour ajouter les millisecondes aux dates des intervalles
  - `--out_dir=<path> pour définir le chemin de destination des time tables de sortie`
  - `--out_type=<type> with type in (ASCII, VO, Internal) pour définir le type de la time table en sortie`
- TT la liste des chemins vers les time tables séparés par une virgule

A défaut d'options :

- ✓ les dates des intervalles ne contiennent pas les millisecondes,
- ✓ la time table est créée dans le répertoire courant,
- ✓ le type de sortie est au format interne AMDA.

5.3.4.5 Formats des TimeTable

5.3.4.5.1 Format ASCII

Le format ASCII est un format texte débutant par plusieurs lignes de commentaire suivies de N lignes chacune comprenant la date de début et la date de fin d'un intervalle de temps.

Exemple :

```
#Time Table generated by AMDA @ CDPP;
#Name: testTable_M_lll;
#Description: ;
#Historic: Union between testTable lll ;
#Creation Date :  2012-09-05T19:15:47;
#
2001-02-07T18:39:11 2001-02-07T18:59:11
2001-02-07T19:49:11 2001-02-07T19:59:11
2001-02-08T02:29:11 2001-02-08T03:04:11
2001-02-08T03:44:11 2001-02-08T03:49:11
```

```
2001-12-23T19:06:00 2001-12-24T13:44:58
2001-12-24T13:54:58 2001-12-24T14:04:58
2001-12-24T19:59:47 2001-12-25T10:56:45
2001-12-25T11:01:45 2001-12-25T13:31:45
2001-12-25T13:41:45 2001-12-25T15:01:45
```

Toute ligne commençant par le caractère # est parcourue pour y trouver un des mots clé suivants (sans contrôle de casse) :

- ✓ #<espace>***Name**<espace>*:**
- ✓ #<espace>***Historic**<espace>*:**
- ✓ #<espace>***Creation**<espace>***Date**<espace>*:**

Ce qui suit ces mots clé est affecté à l'attribut relatif dans la TimeTable à savoir, respectivement, TimeTable::name, TimeTable::history, Timetable::creationDate.
Lorsque la ligne ne contient aucun de ces mots clé, elle est ajoutée à la description de la TimeTable.
Remarque : si la date ne peut être parsée correctement, elle est également ajoutée à la description.

La lecture et l'écriture des fichiers Time Table au format ASCII sont réalisées respectivement par les classes `AsciiTimeTableReader` et `AsciiTimeTableWriter`.

5.3.4.5.2 Format VOTable

Il s'agit d'un format XML, décrit à l'adresse http://www.ivoa.net/documents/VOTable/20130315/PR-VOTable-1.3-20130315.html#XML-schema.

Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<VOTABLE version="1.3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.ivoa.net/xml/VOTable/
v1.3" xmlns:stc="http://www.ivoa.net/xml/STC/v1.30">
 <DESCRIPTION>
        Time Table generated by AMDA @ CDPP;
        Name: myTestTable;
        Description: test;
        Historic: ;
        Creation Date :  2013-04-09T14:36:35;
 </DESCRIPTION>
  <RESOURCE>
   <DESCRIPTION>AMDA @ CDPP</DESCRIPTION>
   <TABLE>
   <FIELD datatype="char" name="Start Time" ID="TimeIntervalStart" ucd="time.start"><DESCRIPTION>time tag for
beginning of interval</DESCRIPTION></FIELD>
   <FIELD datatype="char" name="Stop Time" ID="TimeIntervalStop" ucd="time.stop"><DESCRIPTION>time tag for end of
interval</DESCRIPTION></FIELD>
   <DATA>
   <TABLEDATA>
   <TR><TD>2001-12-24T19:59:47</TD><TD>2001-12-25T09:56:59</TD></TR>
   <TR><TD>2001-12-23T22:27:31</TD><TD>2001-12-24T10:09:18</TD></TR>
   <TR><TD>2001-12-23T22:27:31</TD><TD>2001-12-24T10:09:18</TD></TR>
   <TR><TD>2001-12-23T19:06:00</TD><TD>2001-12-24T12:39:52</TD></TR>
   <TR><TD>2001-12-23T19:06:00</TD><TD>2001-12-24T12:39:52</TD></TR>
   <TR><TD>2001-12-23T19:06:00</TD><TD>2001-12-24T12:39:52</TD></TR>
   <TR><TD>2001-12-23T19:06:00</TD><TD>2001-12-24T12:39:52</TD></TR>
   </TABLEDATA>
   </DATA>
   </TABLE>
   </RESOURCE>
```

L'élément `VOTABLE/DESCRIPTION` est parcouru pour y extraire les informations suivantes :

- ✓ TimeTable::name (pattern <espace>***Name**<espace>*\n)
- ✓ TimeTable::historic (pattern <espace>***Historic**<espace>*\n)
- ✓ TimeTable ::creationDate (pattern <espace>***Creation**<espace>***Date**<espace>*\n**)**

Les autres informations éventuelles contenue dans le texte de cet élément sont perdues.
L'attribut TimeTable::description est quant à lui renseigné avec le contenu de l'élément `VOTABLE/RESOURCE/DESCRIPTION`.

Les intervalles sont extraits depuis les éléments `VOTABLE/RESOURCE/DATA/TABLEDATA/TR/TD`.

La lecture et l'écriture des fichiers Time Table au format VOTable sont réalisées respectivement par les classes `VOTimeTableReader` et `VOTimeTableWriter` et exploitent les fonctionnalités de la librairie déjà utilisée pour la lecture des requêtes : libxml2.

5.3.4.5.3 Format interne AMDA

Il s'agit également d'un format XML, ci-dessous, un exemple de time table au format interne AMDA :

```
<?xml version="1.0"?>
<timetable>
  <name>FTE_c3</name>
  <created>2013-07-14T09:09:32</created>
  <description>Uploaded Time Table
Time Table generated by AMDA @ CDPP;Description: FTE list from Cluster 3 data. From \"A new multivariate time series data analysis
technique: Automated detection of flux transfer events using Cluster data\" by Karimabadi et al., JOURNAL OF GEOPHYSICAL RESEARCH, VOL.
114, A06216, doi:10.1029/2009JA014202, 2009 http://www.agu.org/journals/ja/ja0906/2009JA014202/The list is available as Auxiliary
material \"Data Set S3\"Transformation into AMDA Time Table by V. Genot, CESR, Toulouse, France 29/06/2009 : - millisec have been
omitted - the original event corresponds to the StartTime of the Time Table - if StopTime-StartTime = 1 sec then the event is a
magnetosheath FTE - if StopTime-StartTime = 2 sec then the event is a magnetospheric FTE;Source: Upload Time Table;Creation Date :
2009-07-01 17:16:46 shared by Vincent Genot on 2009-11-24 18:52:50;</description>
  <history>created from another TT</history>
  <nbIntervals>738</nbIntervals>
  <intervals>
    <start>2001-02-02T16:27:12</start>
    <stop>2001-02-02T16:27:13</stop>
  </intervals>
  <intervals>
    <start>2001-02-02T16:39:57</start>
    <stop>2001-02-02T16:39:58</stop>
  </intervals>
  <intervals>
    <start>2001-02-02T17:29:29</start>
    <stop>2001-02-02T17:29:30</stop>
  </intervals>
</timetable>
```

Les métadonnées de la table d'événements sont extraites de la façon suivante :

- ✓ TimeTable::name
  - ⇨ élément `timetable/name`
- ✓ TimeTable::historic
  - ⇨ élément `timetable/history`
- ✓ TimeTable ::creationDate
  - ⇨ élément `timetable/created`
- ✓ TimeTable ::description
  - ⇨ élement `timetable/description`

Les intervalles sont extraits depuis les éléments `timetable/intervals/start` et `timetable/intervals/stop`.

La lecture et l'écriture des fichiers Time Table au format XML interne à AMDA sont réalisées respectivement par les classes `InternalXMLTableReader` et `InternalXMLTableWriter` et, à l'instar des classes de lecture/écriture du format VOTable, exploitent les fonctionnalités de la librairie libxml2.

5.3.4.6 Synthèse du composant

Le diagramme ci-dessous présente les principales classes du composant Time Table :

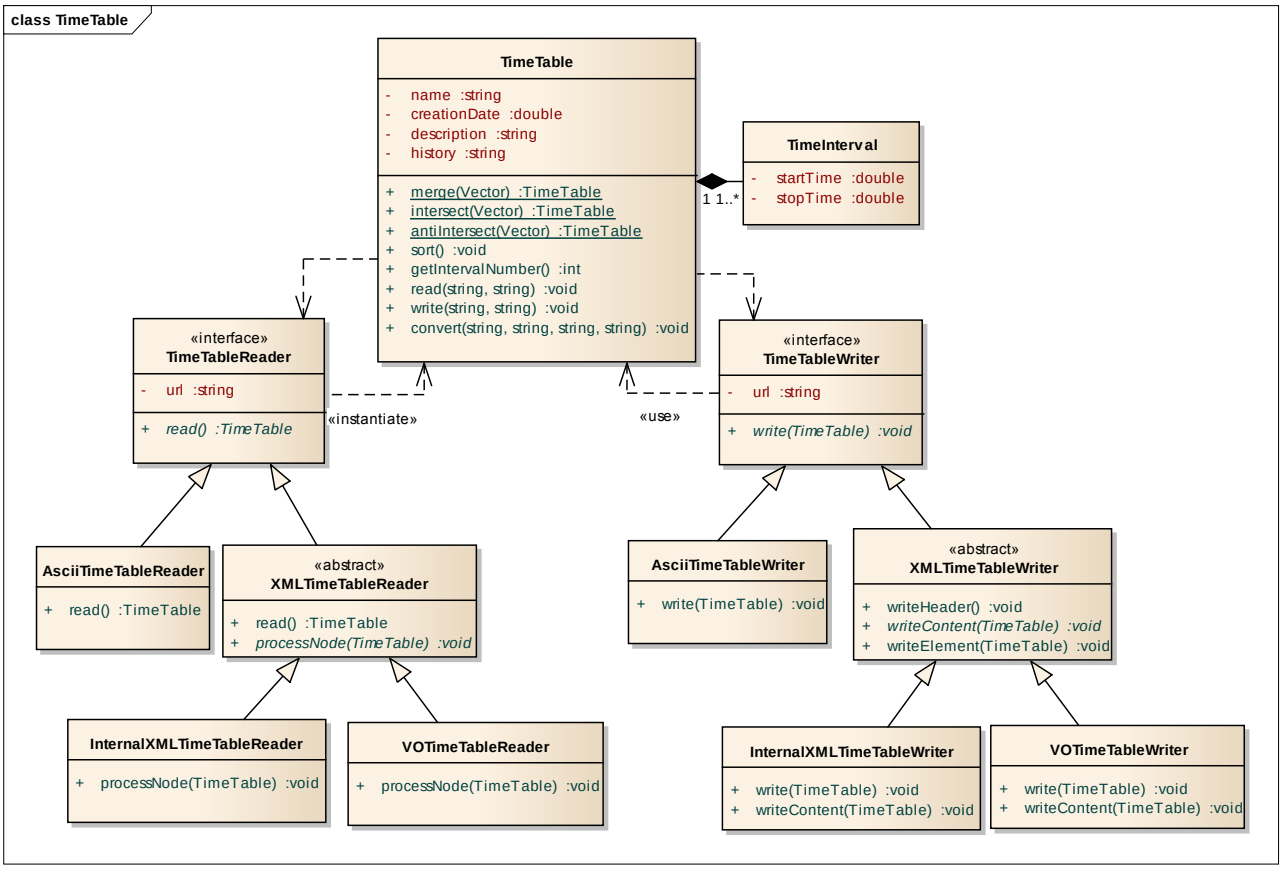


Figure 14 : Classes du composant TimeTable

Une time table est identifiée par un chemin absolu ou relatif (ex : tt.xml, ../tt.xml, d:/tt.xml) ou par une URL de type http(s)://path/to/tt/tt.xml.

Des méthodes `read(url, format)` et `write(url, format)` dans l'objet `TimeTable` factorisent les codes de création de reader/writer et permettent de s'abstraire de leur utilisation dans le code appelant. Dans la même idée, la méthode `convert (fromUrl, fromFormat, toUrl, toFormat)` factorise la séquence `read/write` pour une conversion directe d'une time table d'un format donné en un autre format.

Des implémentations de `TimeTableReader` et `TableTimeWriter` existent pour chacun des formats connus de Time Table. Le niveau supplémentaire introduit par les classes abstraites `XMLTimeTableReader` et `XMLTimeTableWriter` permet de factoriser les traitements de fichiers XML indépendants de tout format spécifique.

La création de l'implémentation de `TimeTableReader` (resp. `TimeTableWriter`) est effectuée par une fabrique dédiée `TimeTableReaderFactory` (resp. `TimeTableWriterFactory`). Ceci permet de centraliser la création d'objets et de simplifier notamment l'ajout éventuel d'autres formats de time table à l'avenir.

5.3.5 Composant « Data Mining»

5.3.5.1 Exigences techniques

Id	Description	Commentaire

AMDA_DM_010	Le système propose une requête permettant de créer une Time Table à partir de la définition d'une condition.	STB §6.1, US1
AMDA_DM_011	Le format des Time Tables créées peut être : ✓ ASCII ✓ VO Table ✓ Format interne AMDA (XML)	STB §6.3, US1, US43
AMDA_DM_020	Le système propose une requête permettant de créer une Time Table avec des intervalles de temps correspondants à une absence de données. Dans ce cas, la Time Table est préfixée avec « Gaps_ ».	US4
AMDA_DM_030	Lorsque les paramètres d'une expression conditionnelle n'ont pas la même résolution, un ré-échantillonnage doit être effectué.	US1
AMDA_DM_040	Le système propose de définir les intervalles de temps d'une requête à partir de la référence d'une Time Table.	STB §3.1, US5, US6
AMDA_DM_050	Le système doit permettre de définir un format de compression éventuel sur les fichiers résultats d'une requête de datamining : ✓ Tar+gzip ✓ Zip	US31

Tableau 4 : Exigences du composant Data Mining

5.3.5.2 Objet

Le composant Data Mining permet de produire une Time Table à partir d'une recherche conditionnelle (expression). Les intervalles de temps stockés dans la Time Table correspondent au moment où la condition est vérifiée. Une fois la tablé créée, celle-ci est enregistrée dans un fichier au format XML, ASCII ou VO Table. Ce composant dérive de `ParamOutput` et de `VisitorOfParamData` qui permet de donner un accès au résultat fourni par la recherche conditionnelle.
La production d'une Time Table est décrite dans une requête XML.

5.3.5.3 Structure XML

Le format d'une requête XML comprend les paramètres suivants pour caractériser une Time Table:
- ✓ Le format des dates stockées,
- ✓ Le format de sortie du fichier,
- ✓ La structure à utiliser pour créer le fichier,
- ✓ Le nom du fichier
- ✓ L'identifiant du paramètre sur lequel la requête s'applique,

La structure de la requête pour un Data Mining est de la forme suivante :

```
<dataMining>
      <timeFormat>ISO</timeFormat>
      <fileFormat>ASCII</fileFormat>
      <outputStructure>one-file</outputStructure>
      <fileName>tt_result</fileName>
      <param id= "imf"></param>
<dataMining>
```

5.3.5.3.1 Données

Pour un fichier spécifique, le format de stockage des dates est unique et est par défaut le format ISO.

Le format de sortie du fichier peut être de trois types :
- ✓ XML : c'est un format propre à AMDA basé sur une structure XML,
- ✓ VOT : VO Table est un format spécifique basé sur une structure XML,
- ✓ ASCII : le format du fichier est purement textuel

Ces deux informations doivent obligatoirement apparaître dans une requête. Si l'une des deux balises est manquante alors la requête n'est pas prise en compte.

La balise `outputStructure` permet d'indiquer pour quel intervalle de recherche un fichier est créé. Cette balise accepte deux valeurs :
- ✓ `one-file-per-interval` pour laquelle un fichier est créé par intervalle de temps de TimeTable,
- ✓ `one-file` pour laquelle un seul fichier est créé pour tous les intervalles de temps de TimeTable.

Cette balise sert uniquement lorsque l'intervalle de temps d'une requête est défini par un fichier TimeTable (cf. **5.3.5.4.3**). Elle est optionnelle et sa valeur par défaut est `one-file`.

La balise `fileName` permet d'identifier le nom de sortie du fichier généré. Par défaut, un nom de fichier est attribué et est de la forme timeTable_PARAMID_XXX (où PARAMID est le nom du paramètre sur lequel les intervalles sont créés et XXX est la date en secondes à laquelle le fichier a été généré). L'extension du fichier n'est pas à préciser, elle est automatiquement ajoutée.

Le rôle de la balise `param` est d'identifier sur quel paramètre le traitement de création des intervalles doit s'appliquer. L'identifiant du paramètre fait référence à un fichier XML séparé.

5.3.5.3.2 Emplacement

La structure XML du composant DataMining peut être placée à deux endroits :
- ✓ dans le fichier de définition d'une requête XML,
- ✓ dans le fichier de définition d'un paramètre XML

Ci-dessous la structure du composant DataMining insérée dans un fichier de définition d'une requête XML :

```
<request>
      <params>
            <param id="imf_cond"/>
      </params>
      <times>
            <interval>
                  <startTime>2008000000000000</startTime>
                  <timeInterval>0000010000000000</timeInterval>
            </interval>
      </times>
      </outputs>
            <dataMining>
                  <timeFormat>ISO</timeFormat>
                  <fileFormat>ASCII</fileFormat>
                  <outputStructure>one-file</outputStructure>
                  <fileName>tt_result</fileName>
                  <param id="imf_cond"></param>
            <dataMining>
      </outputs>
</request>
```

Et maintenant la voici dans un fichier de définition d'un paramètre XML :

```
<param xml:id="imf_cond">
      <get>
            <amdaParam name="imf" />
      </get>
      <process>greater_than($imf[0],0)</process>
      <output>
            <dataMining>
                  <timeFormat>ISO</timeFormat>
                  <fileFormat>ASCII</fileFormat>
                  <outputStructure>one-file</outputStructure>
                  <fileName>tt_result</fileName>
            <dataMining>
      </output>
</param>
```

La différence entre ces deux structures XML pour le composant DataMining est la disparition de la balise `param`. En effet, dans le fichier XML d'un paramètre celle-ci ne devient plus obligatoire puisque le lien entre le paramètre et la sortie est direct (une seule définition de paramètre par fichier).

La structure du composant DataMining peut donc être définie à la fois dans le fichier d'une requête et d'un paramètre. Lorsque le fichier d'un paramètre contient une telle structure, celle-ci doit être vue comme le comportement à adopter par défaut pour générer le fichier de sortie. En revanche, si cette structure est volontairement ajoutée dans le fichier d'une requête alors qu'il y en a déjà une par défaut alors c'est celle-ci qui doit être prise en compte (ce principe n'est valable que pour le même paramètre).

Pour pallier ce risque de multidéfinition, un niveau de priorité est donc mis en œuvre pour choisir quelle structure doit être utilisée. La structure définie dans le fichier d'un paramètre doit donc avoir une priorité plus faible que celle définie dans le fichier d'une requête.

5.3.5.4 Intervalles

5.3.5.4.1 Condition vérifiée

Un intervalle est caractérisé par une date de début et de fin englobant un ensemble de dates qui vérifient la condition définie dans le fichier d'un paramètre dans la balise `process`. Dès lors que la condition n'est plus vérifiée, l'intervalle s'arrête et un nouvel intervalle ne commence que lorsque la condition est à nouveau valide.

Pour chaque intervalle retenu, un delta est soustrait à la date de début et ajouté à la date de fin. Ce delta correspond à la moitié du temps d'échantillonnage du paramètre.

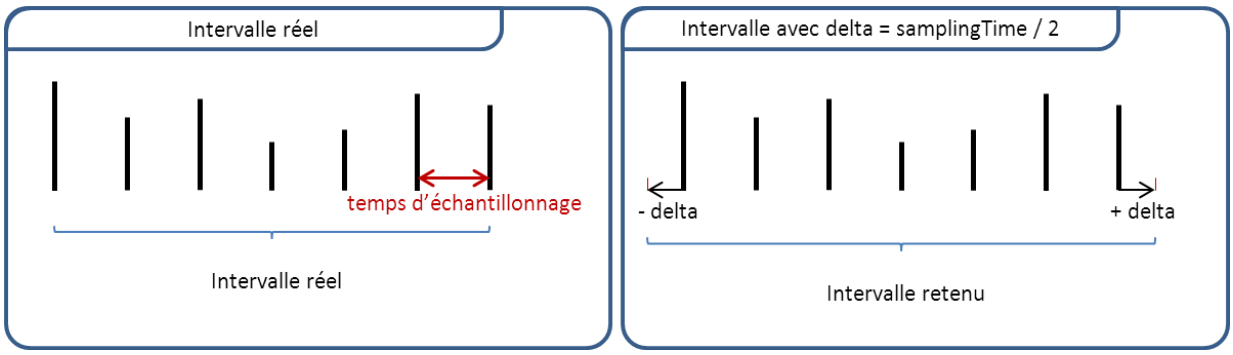


Figure 15 : Définition d'un intervalle

5.3.5.4.2 Trou de données

Pour un paramètre donné, il est possible que celui-ci contienne à certains moments des « trous de données » et cela peut poser problème lors de la génération des intervalles. Pour résoudre en partie celui-ci, une balise `gap_threshold` est insérée dans le fichier de définition d'un paramètre XML qui permettra ainsi de fixer une « tolérance » et donc d'attribuer des valeurs par interpolation ou moyennage.

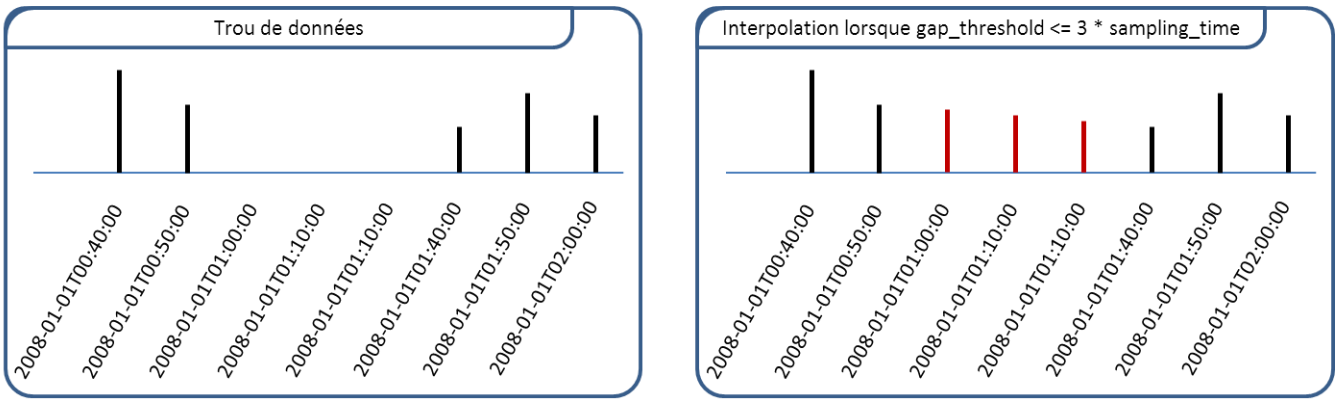


Figure 16 : Interpolation des données

Sur la figure ci-dessus, un trou de données est apparu et celui-ci a été rempli par interpolation (l'interpolation a été uniquement faite dans le cas où le trou de données est inférieur ou égal à trois fois le temps d'échantillonnage). Dans le cas contraire, le trou de données ne sera pas rempli par interpolation.

Cette opération peut être très utile dès lors qu'un ré échantillonnage se produit (i.e. lorsque l'échantillonnage d'origine n'est pas un facteur de celui demandé).

Lorsque le trou de données est supérieur à `gap_threshold * sampling_time`, alors cette intervalle est stocké dans un fichier séparé dont le nom est préfixé par « Gaps_ ».

5.3.5.4.3 Intervalles d'entrées

cf **5.3.2**

5.3.5.4.4 Intervalles trop courts

Lorsqu'un fichier time table est utilisé en entrée d'une requête, il est possible que certains intervalles de temps qui le composent soient trop courts. Dans ce cas-là, l'intervalle de temps est strictement plus petit que le temps d'échantillonnage et ne doit pas être traité.

Ces intervalles de temps trop courts sont stockés dans un fichier préfixé par « TooSmall_ ».

5.3.5.5 Dépendances

Le composant Data Mining exploite les Time Table du composant éponyme. Pour pouvoir traiter la requête XML, il s'appuie sur le composant `ParamOutput` qui définit le squelette pour le traitement d'une sortie.

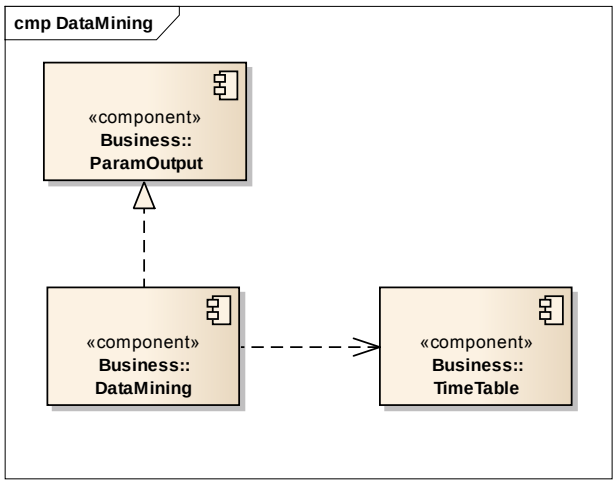


Figure 17 : Dépendances du composant Data Mining

5.3.5.6 Classe Data Mining

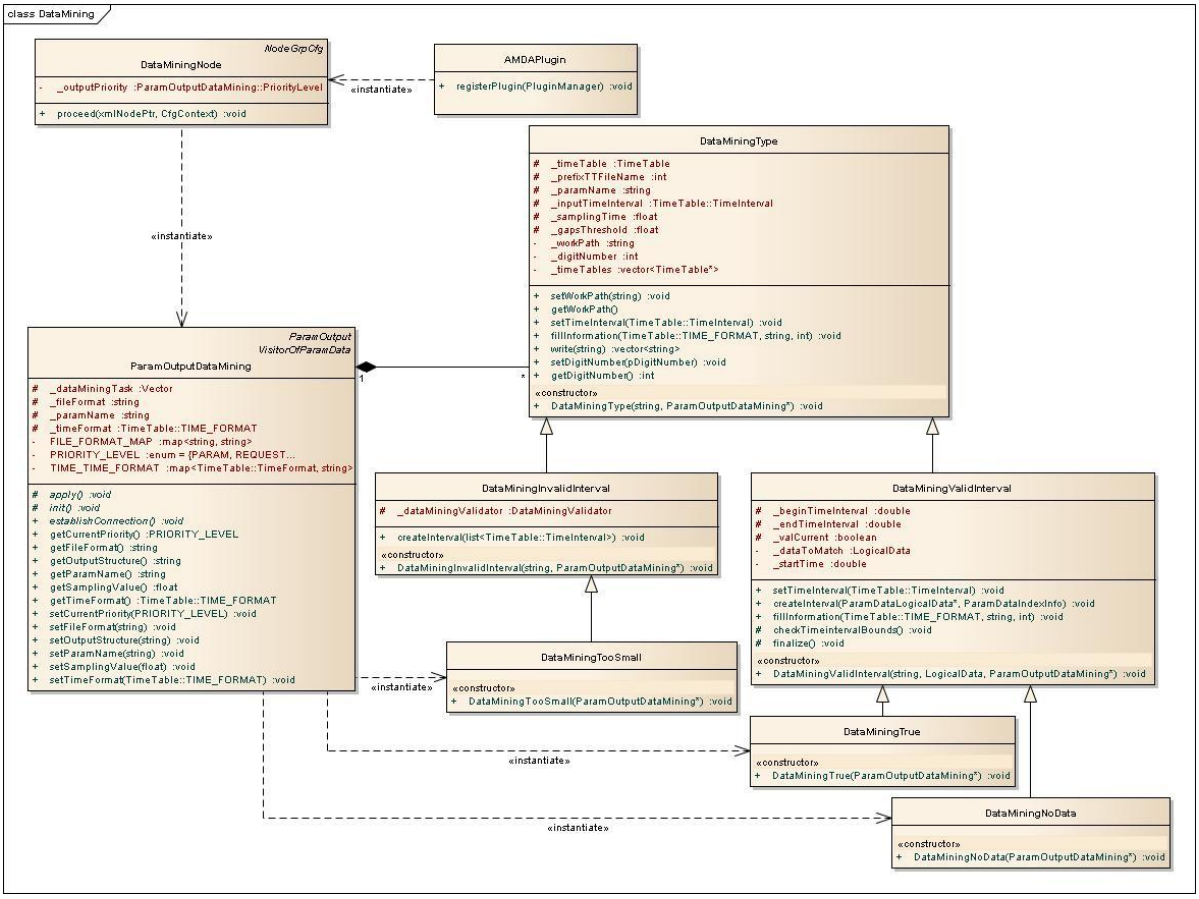


Figure 18 : Classes du composant DataMining

La création de ce composant nécessite de définir le schéma XSD de DataMining dans `config/DataBaseParameters/xsd/request/dataMining.xsd`. Ce fichier est également notifié dans le fichier « all.xsd » au même niveau que le précédent.

La classe AMDAPLugin permet de définir quelle partie du fichier XML est contrôlée par le composant DataMining. Ici se sera :

- ✓ request/outputs/dataMining
- ✓ param/output/dataMining

Pour que le parseur XML soit capable de donner la main au composant DataMining sur sa structure XML lorsque celle-ci est insérée dans le fichier de définition d'un paramètre XML, il faut modifier au préalable le corps du constructeur AMDA::XMLParameterConfigurator::ParamNode::ParamNode() pour que celui-ci définisse un nœud sur la balise output à l'aide d'une instance de type AMDA::XMLConfigurator::NodeCfgSPtr.

La classe AMDAPlugin sera également en charge de définir le niveau de priorité des deux chemins puisque ce n'est seulement qu'à cet endroit que la différence peut être faite entre les deux. Le niveau de priorité sera transmis à la classe ParamOutputDataMining par le biais de la classe DataMiningNode qui est instanciée dans la classe AMDAPlugin.

Etant donné que le mécanisme de ré échantillonnage avec prise en compte du gap_threshold est déjà implémenté, il s'agit simplement de mettre à jour le schéma du fichier XSD lié au paramètre « parameter.xsd » mais aussi de transmettre l'information à travers les différentes classes du noyau. Pour cela, ajouter l'attribut gap_threshold et ses accesseurs dans la classe AMDA::Parameters::Parameter. Définir également la constante par défaut DEFAULT_GAP_THRESHOLD_VALUE du gap qui vaut 5, puis modifier la fonction AMDA::Parameters::ProcessNode::proceed(xmlNodePtr,AMDA::Parameters::CfgContext&) ainsi que la définition de la fonction injectResamplingIntoProcess(const double&, const char*) pour prendre en compte cette valeur lorsqu'un ré échantillonnage est effectué.
Rajouter également un nœud dans la classe AMDA::XMLParameterConfigurator::ParamNode::ParamNode() pour que la balise gap_threshold soit lue.

La classe DataMiningNode fait le lien entre le type de sortie et la classe à utiliser pour répondre à la requête de sortie. Elle hérite donc de la classe NodeGrpCfg et définit la fonction proceed pour créer une instance de ParamOutputDataMining. C'est cette classe « nœud » qui sera en charge de lire toute les balises de la structure XML et de les transmettre à l'instance ParamOutputDataMining.

La classe de base ParamOutputDataMining permet successivement d'initialiser la classe, d'établir une connexion avec les différents paramètres dont elle a besoin et enfin de traiter la requête de sortie.
Etant donné que le composant DataMining peut avoir différents comportements (intervalle trop petit par rapport à ce qui est demandé, trou de donnée, …) ce n'est pas directement cette classe qui établit une connexion avec le(s) paramètre(s) concerné(s) et qui crée les intervalles mais la classe DataMiningType. Ainsi, à chaque intervalle d'entrée, est associée une ou plusieurs instances de DataMiningType.

La classe DataMiningType se généralise ensuite en deux grandes parties :

- ✓ DataMiningInvalidInterval qui est chargé d'effectuer des opérations sur les intervalles d'entrée qui ont un délai plus petit que le temps d'échantillonnage.
- ✓ DataMiningValidInterval qui est chargé d'effectuer des opérations sur les intervalles dits « valide » (le délai de ces intervalles est supérieur ou égal au temps d'échantillonnage).

Les classes dérivant de DataMiningInvalidInterval sont des cas particuliers qui ne s'appliquent pas directement sur un intervalle de temps précis mais sur tous les intervalles invalides. La classe qui en

découle est DataMiningTooSmall qui a pour unique but d'écrire dans une TimeTable ces intervalles trop courts.
Etant donné qu'il s'agit d'un principe pouvant s'appliquer sur d'autres composants, l'implémentation du mécanisme de traitement des intervalles de temps doit se situer au début de la méthode init implémenté par la classe ParamOutput. Ce n'est qu'à ce moment précis que toutes les informations nécessaires au processus de traitement sont disponibles. Pour ce faire, le temps d'échantillonnage doit être récupéré au niveau du paramètre sur lequel la connexion a été établie (phase amont establishConnection). Si cette valeur est nulle, alors il s'agit de descendre dans l'arborescence des paramètres jusqu'à trouver une valeur non nulle.

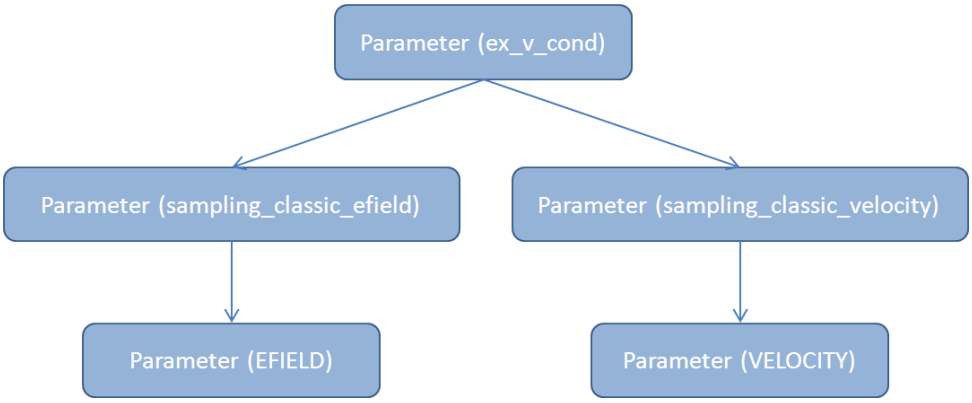


Figure 19 : Chaînage de paramètre

Dans la figure ci-dessus, chaque paramètre est lu jusqu'à trouver une valeur non nulle. Dans le cas d'une divergence de paramètre, seul le temps d'échantillonnage qui a la plus petite valeur sera retenue. Si toutefois la fin d'une branche (feuille) est atteinte (i.e. EFIELD), alors une requête est envoyée au serveur pour récupérer la valeur « MinSampling » de ce paramètre. Ce mécanisme procède par récursivité et remonte la valeur du temps d'échantillonnage.
Pour que ce principe soit possible, il faut rajouter une ligne dans les classes ParameterCreatorFromExpression et ProcessStandard pour faire le lien entre les deux paramètres ParameterMananager ::addParameter(Parameter, String, Parameter).

De cette manière, la classe ParamOutput contient deux listes d'intervalles de temps :
- ✓ Une liste contenant tous les intervalles de temps strictement inférieurs au temps d'échantillonnage,
- ✓ Une liste contenant tous les intervalles de temps supérieurs ou égaux au temps d'échantillonnage.

A l'inverse, les classes DataMiningTrue et DataMiningNoData récupèrent le résultat de la condition à différents instants t compris dans un intervalle valide. DataMiningTrue traite sur une condition qui doit être vérifiée à l'inverse de DataMiningNoData qui opère sur une condition qui ne peut être vérifiée et dont le délai pendant lequel la condition ne peut être vérifiée est supérieur ou égal à gap_threshold * sampling_value (sampling_value est le temps d'échantillonnage de base du paramètre et non celui de ré-échantillonnage).

Le contrôle sur l'écriture de la ou des TimeTable(s) est detenu par la classe DataMiningType. Dans le cas où la balise outputStructure est positionnée sur la valeur one-file-per-interval, il est de la responsabilité de la classe ParamOutputDataMining d'indiquer à quel moment une TimeTable doit être écrite dans un fichier (appel de la méthode writeTTContent). De même chaque fichier devra être suffixé par un index (i.e. 0001, 0002, 0003, ….), par défaut l'index sera affiché sur 4 digits mais ce nombre peut être amené à évoluer automatiquement suivant le nombre de fichier à créer.

Le diagramme d'activité ci-dessous représente le mécanisme de création des intervalles :

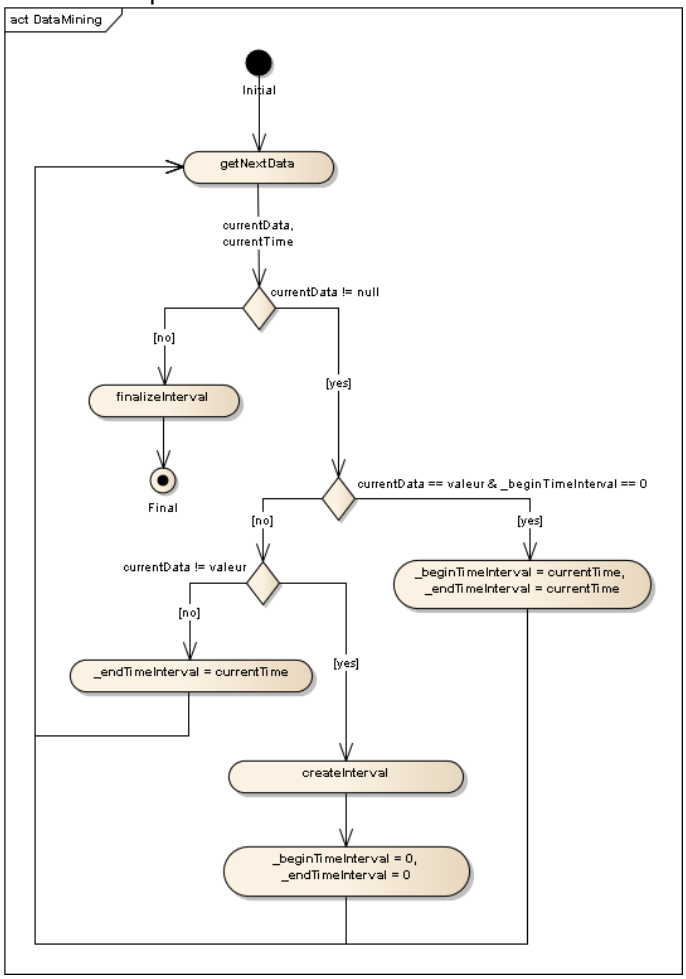


Figure 20 : Diagramme d'activité de création des intervalles

Lors de la création d'un intervalle plusieurs cas sont à prendre en compte :

- ✓ Si la date de début du premier intervalle concorde avec celle de la date de départ de recherche alors il ne doit pas lui être soustrait le temps d'échantillonnage divisé par deux,
- ✓ Si la date de fin du dernier intervalle concorde avec celle de la date de fin de recherche alors il ne doit pas lui être ajouté le temps d'échantillonnage divisé par deux,
- ✓ Si une seule date vérifie la condition alors l'intervalle devient [currentTime – samplingTime/2, currentTime + samplingTime/2] et doit respecter les deux cas cités ci-dessus.

Dans certains cas, il est possible qu'il soit nécessaire de finaliser un intervalle. Celui-ci se produit à chaque fois que la dernière date du résultat de la recherche vérifie la condition. Ainsi, l'exécution se terminera en laissant en suspend un intervalle en cours de création. Pour pallier ce problème, il suffira uniquement, pour ce contexte précis, de vérifier si un intervalle est en cours de création et donc de donner comme date de fin de cet intervalle la date de fin de la recherche (méthode `finalize()`).

5.3.6 Composant « PostProcessing»

5.3.6.1 Exigences techniques

Id	Description	Commentaire
AMDA_PP_010	Le système doit permettre d'associer, si besoin, un ou plusieurs post-traitements à un type d'output.	STB §4,US30

Tableau 5 : Exigences du composant PostProcessing

5.3.6.2 Objet

Le composant PostProcessing est responsable des éventuels traitements sur les résultats d'une requête. Chaque type de résultat (implémentation de `ParamOutput`) nécessite des post-traitements différents.

5.3.6.3 Requête XML

Il existe désormais la possibilité de définir une liste ordonnée de post-traitements pour chacune des implémentations de `ParamOutput`. Cette liste est contenue dans un noeud `postProcess` optionnel pour tout output (download, datamining etc.) et est représentée par une séquence de nœuds ; chacun dédié à un post-traitement spécifique.

Exemple de requête pour un post-traitement *tar+gzip* sur le résultat d'une requête de *download* :

```
<outputs>
    <download>
        <timeFormat>ISO</timeFormat>
        <param id='velocity'/>
        <postProcess>
            <tar/>
            <gzip/>
         </postProcess>
    </download>
</outputs>
```

5.3.6.4 Classes du package

5.3.6.4.1 Classes PostProcessing et ParamOutput

Le diagramme ci-dessous présente un sous-ensemble des classes du package `PostProcessing` et propose une illustration des implémentations possibles de post-traitements pour l'output Download (classes `ZipPostProcessing`, `GzipPostProcessing` et `TarPostProcessing`).

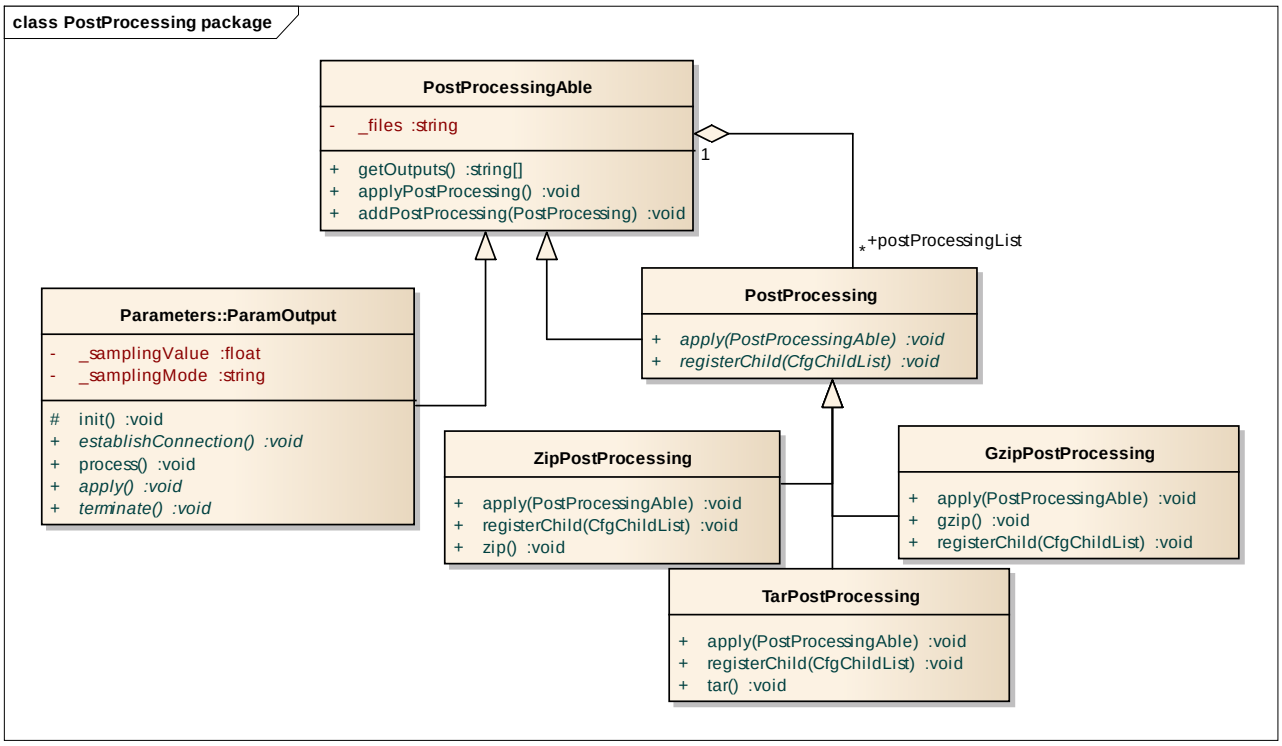


Figure 21 : Classes du package postProcessing

Tout élément pouvant donner lieu à un post-traitement implémente une même classe abstraite `PostProcessingAble`. Un `PostProcessingAble` est associé à une liste de post-traitements qui peut être vide. Cette classe présente une méthode `applyPostProcessing()` non virtuelle qui parcourt cette liste et lance les post-traitements.
Dans le cas présent, il est possible d'associer un ou plusieurs post-traitements à toute implémentation de `ParamOutput`.
Par ailleurs, les post-traitements doivent avoir accès à certaines données des implémentations de `ParamOutput` – par exemple, le post-traitement *zip* doit pouvoir récupérer la liste des fichiers à compresser. C'est également le rôle de la classe `PostProcessingAble` de regrouper l'ensemble des accès aux données nécessaires aux post-traitements. Les types d'output donnant lieu à des éventuels post-traitements doivent surcharger les méthodes qui les concernent – des implémentations par défaut sont proposées par `PostProcessingAble`. Il est ensuite de la responsabilité de chaque post-traitement de s'assurer que la ou les données requises sont bien fournies avant de s'exécuter. Ainsi, les post-traitements peuvent être exécutés directement par la classe abstraite `PostProcessingAble`.
Cette dernière pourra être complétée au besoin lors de l'ajout de nouveaux types de post-traitements.

Enfin, chaque post-traitement implémente une interface `PostProcessing` et ses méthodes :
- ✓ `apply(PostProcessingAble)` dont le contrat est de réaliser le post-traitement.
- ✓ `registerChildList()` dont le contrat est de définir le nœud du post-traitement et le traitement à réaliser.

Cette interface hérite de `PostProcessingAble` pour permettre de chaîner les traitements (cas du tar + gzip par exemple) : il est ainsi possible d'appliquer un post-traitement sur un post-traitement.

5.3.6.4.2 Traitement du nœud <postProcess>

Il existe des classes dédiées au traitement de chacun des nœuds impliqués dans le post-traitement à savoir :

- ✓ `PostProcessingNode` qui traite le contenu du nœud `postProcess` et contient la liste des nœuds enfants possibles complétée dynamiquement par appel aux implémentations de la méthode `registerChildList()` de chacun des post-traitements.
- ✓ `xxxNode` (ex : `ZipNode`, `TarNode` etc.) pour chacun des post-traitements ; la méthode `proceed()` a ici pour rôle d'ajouter le post-traitement associé à la liste des post-traitements de `ParamOutput`.

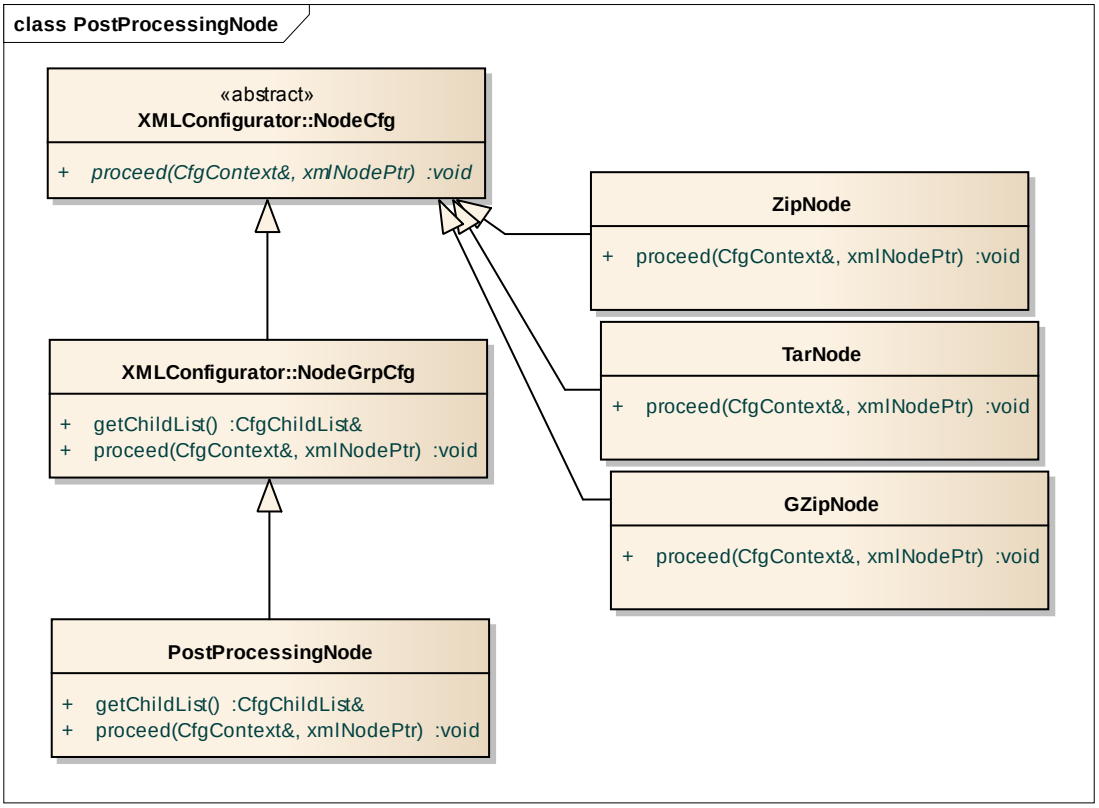


Figure 22 : Classes de lecture des noeuds de post-traitement

5.3.6.4.3 Classe PostProcessingRegistry

Chacun des post-traitements doit être répertorié dans un registre porté par la classe `PostProcessingRegistry`. Cet enregistrement est réalisé de manière statique par chacun des post-traitements.

5.3.6.5 Dynamique

La réalisation d'un post-traitement pour un output donné requiert les étapes suivantes:

- ✓ Lecture de la requête XML et ajout du post-traitement à la liste ordonnée des post-traitements de `ParamOutput`
- ✓ Exécution du post-traitement après traitement de l'output

5.3.6.5.1 Lecture de la requête XML

Lorsqu'un nœud `zip` est rencontré sous le nœud `postProcess` d'un output, le traitement du nœud est invoqué via la méthode `proceed()`. Ce traitement crée une instance du post-traitement associé (`ZipPostProcessing`) et l'ajoute au `ParamOutput` courant du contexte.

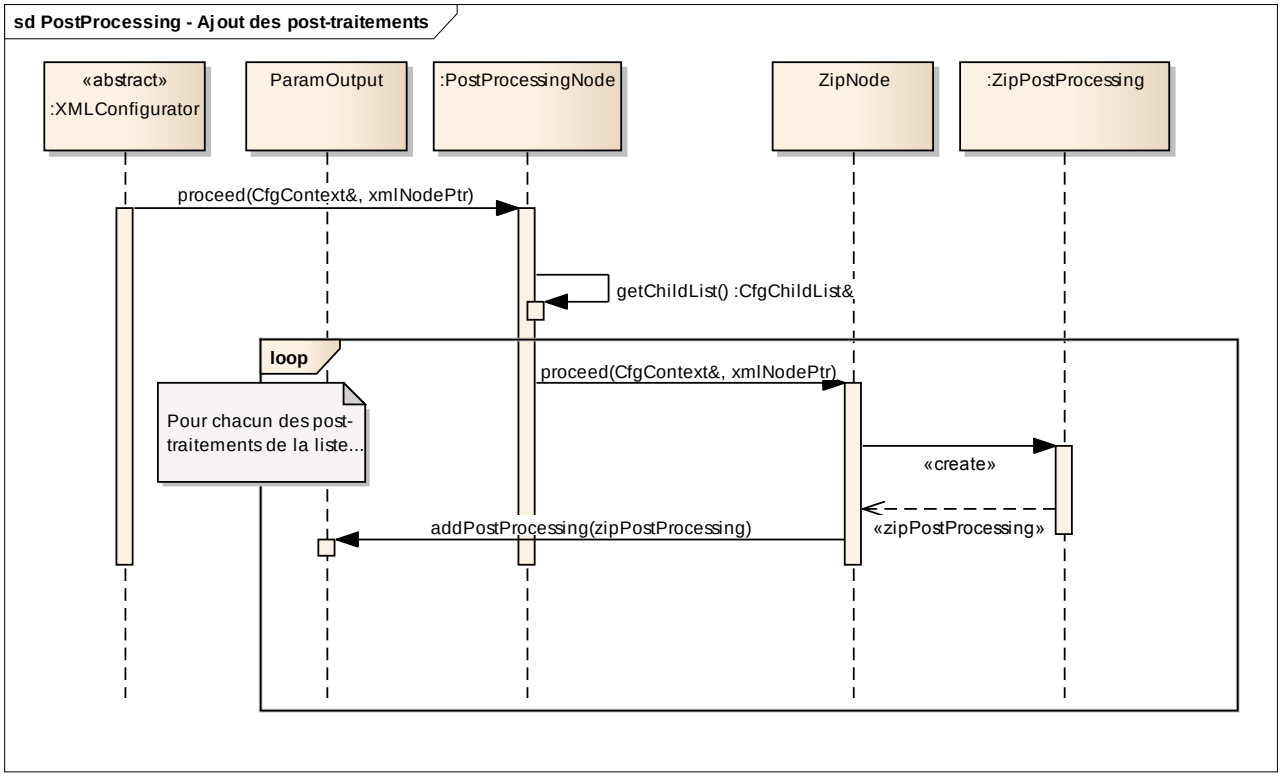


Figure 23 : Scenario "Lecture d'une requête XML – noeud postProcess"

5.3.6.5.2 Exécution des post-traitements

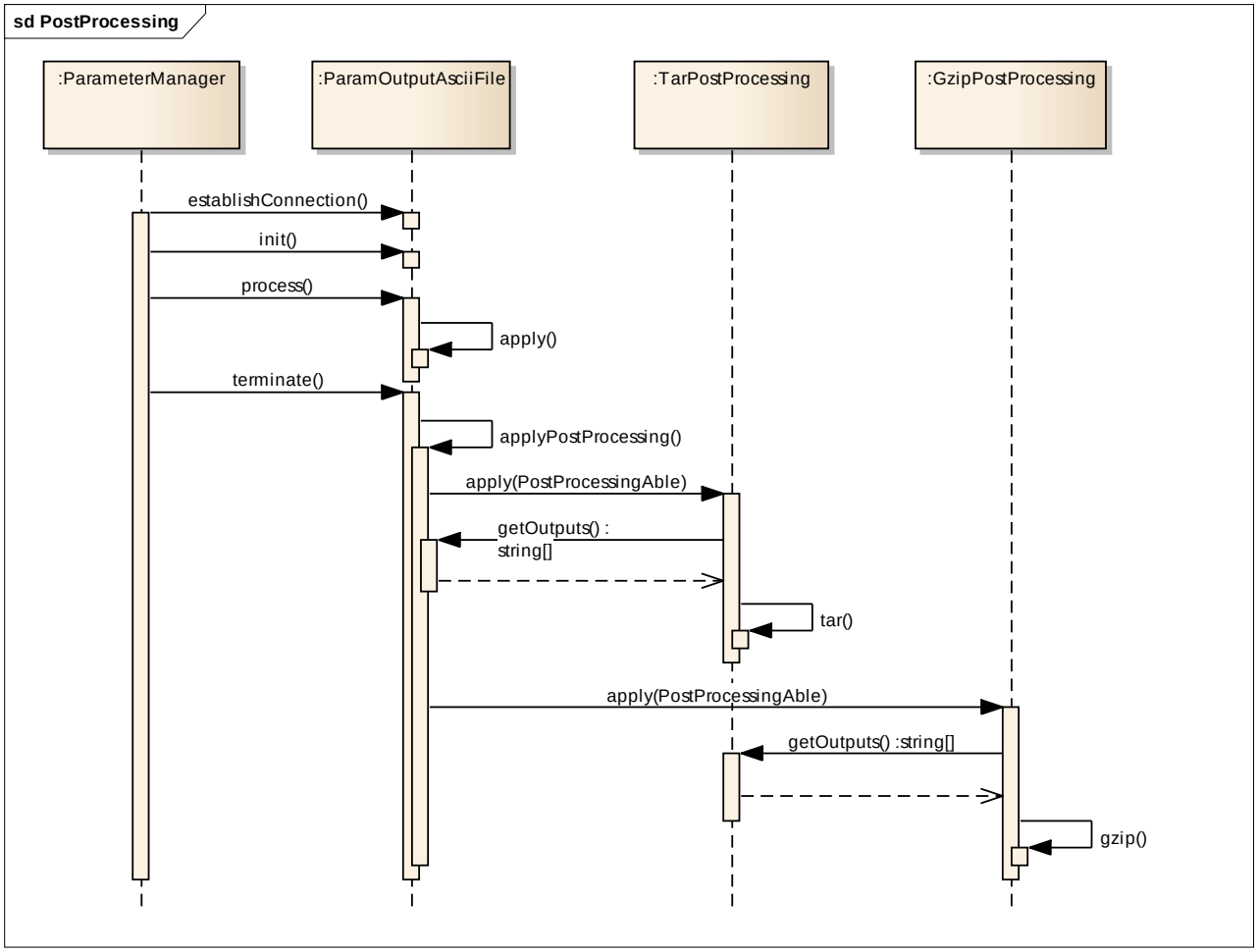


Figure 24 : Scenario "Exécution d'un post-traitement"

Après l'exécution de l'output (`apply()`), le post-traitement est exécuté à son tour (`terminate()` puis `applyPostProcessing()`). L'exécution du post-traitement peut, si nécessaire, récupérer des données de l'output via la classe `PostProcessingAble` (ici `getOutputs()`) et exécuter sa tâche. Lorsque la liste des post-traitements contient plusieurs post-traitements, ceux-ci sont « chainés » : le second post-traitement est réalisé sur le résultat du post-traitement précédent – appel de `TarPostProcessing::getOutputs`.

5.3.6.6 Ajout d'un nouveau post-traitement

En synthèse, l'ajout d'un nouveau post traitement doit suivre les étapes suivantes :
1/ Ajout du nœud `<new-post-process>` dans le fichier descriptif de requête `postProcessing.xsd`
2/ Implémentation `<new-post-process>Node` de la classe `XMLConfigurator::NodeCfg` pour lire le nœud `<new-post-process>`
3/ Implémentation de l'interface `PostProcessing`, méthodes `apply()` et `registerChildList()`
4/ Enregistrement statique de la classe d'implémentation de `PostProcessing` auprès de `PostProcessingRegistry`.
5/ [Optionnel] Complétion de la classe `PostProcessingAble` si le post-traitement nécessite des informations particulières sur l'output

Des exemples pour les étapes ci-dessus sont présentés en annexe 1. A noter qu'un nouveau post-traitement peut-être implémenté dans un plugin.

5.3.7 Composant « Download»

5.3.7.1 Exigences techniques

Id	Description	Commentaire
AMDA_DL_010	Le système doit permettre une requête de download à partir des intervalles d'une Time Table	STB §5.1, US5, US9, US10, US11, US12, US13
AMDA_DL_020	Le système doit gérer une option de sortie pour les requêtes de download sur une Time Table : ✓ Un fichier différent par intervalle de temps ✓ Un fichier unique concaténant chaque intervalle de temps	STB §5.1, US9, US10, US11
AMDA_DL_021	Le système doit permettre une requête de download sur plusieurs paramètres dans un même fichier.	US8
AMDA_DL_030	Le système doit permettre de définir un format de compression éventuel sur les fichiers résultats d'une requête de download : ✓ Tar+gzip ✓ Zip	STB §5.2, US30

Tableau 6 : Exigences du composant Download

5.3.7.2 Objet

Le composant Download existant est complété pour :
- ✓ traiter une requête de download sur plusieurs paramètres,
- ✓ traiter une requête de download sur une liste d'intervalles fournie par une time table,
- ✓ générer un fichier par intervalle ou un fichier pour tous les intervalles,
- ✓ autoriser un ou plusieurs post-traitements (zip, tar, gzip) sur le résultat de la requête.

5.3.7.3 Requête XML

Il est possible de définir, pour un output de type Download, une liste de paramètres en multipliant les balises `<param>`, l'output produit alors un fichier contenant les valeurs de ces deux paramètres. Lorsque ces paramètres ne proviennent pas d'un même jeu de données, la résolution à adopter est précisée dans une balise `<timeResolution>`.
Lorsque la requête est réalisée à partir d'une liste d'intervalles (timetable), la balise `<outputStructure>` peut prendre les valeurs suivantes :
- ✓ `one-file` (par défaut) : génération d'un fichier pour tous les paramètres et pour tous les intervalles
- ✓ `one-file-per-interval` : génération d'un fichier pour tous les paramètres par intervalle

Exemple de requête pour un download de deux paramètres dans un même fichier avec ré-échantillonnage (ici le nom du fichier de sortie est spécifié dans la balise optionnelle `<fileName>`) :

```
<outputs>
    <download>
        <timeFormat>ISO</timeFormat>
        <fileFormat>ASCII</fileFormat>
```

```
        <fileName>download_result</fileName>
        <param id='imf'/>
        <param id='dst'/>
        <timeResolution>600</timeResolution>
        <outputStructure>one-file</outputStructure>
    </download>
</outputs>
```

Exemple de requête pour un download de deux paramètres dans deux fichiers distincts :

```
<outputs>
    <download>
        <timeFormat>ISO</timeFormat>
        <fileFormat>ASCII</fileFormat>
        <param id='dst'/>
        <outputStructure>one-file</outputStructure>
    </download>
    <download>
        <timeFormat>ISO</timeFormat>
        <fileFormat>ASCII</fileFormat>
        <param id='imf'/>
        <outputStructure>one-file</outputStructure>
    </download>
</outputs>
```

5.3.7.4 Classes du package

Le diagramme ci-dessous présente les classes principales du package `Download`. Des modifications sont apportées à l'output `ParamOutputAsciiFile` pour prendre en compte non plus un paramètre mais une liste de paramètres stockés dans une structure de type map(nom de paramètre, paramètre). Les méthodes `setParam` et `setParameter` sont remplacées respectivement par les méthodes `addParam` et `addParameter`.

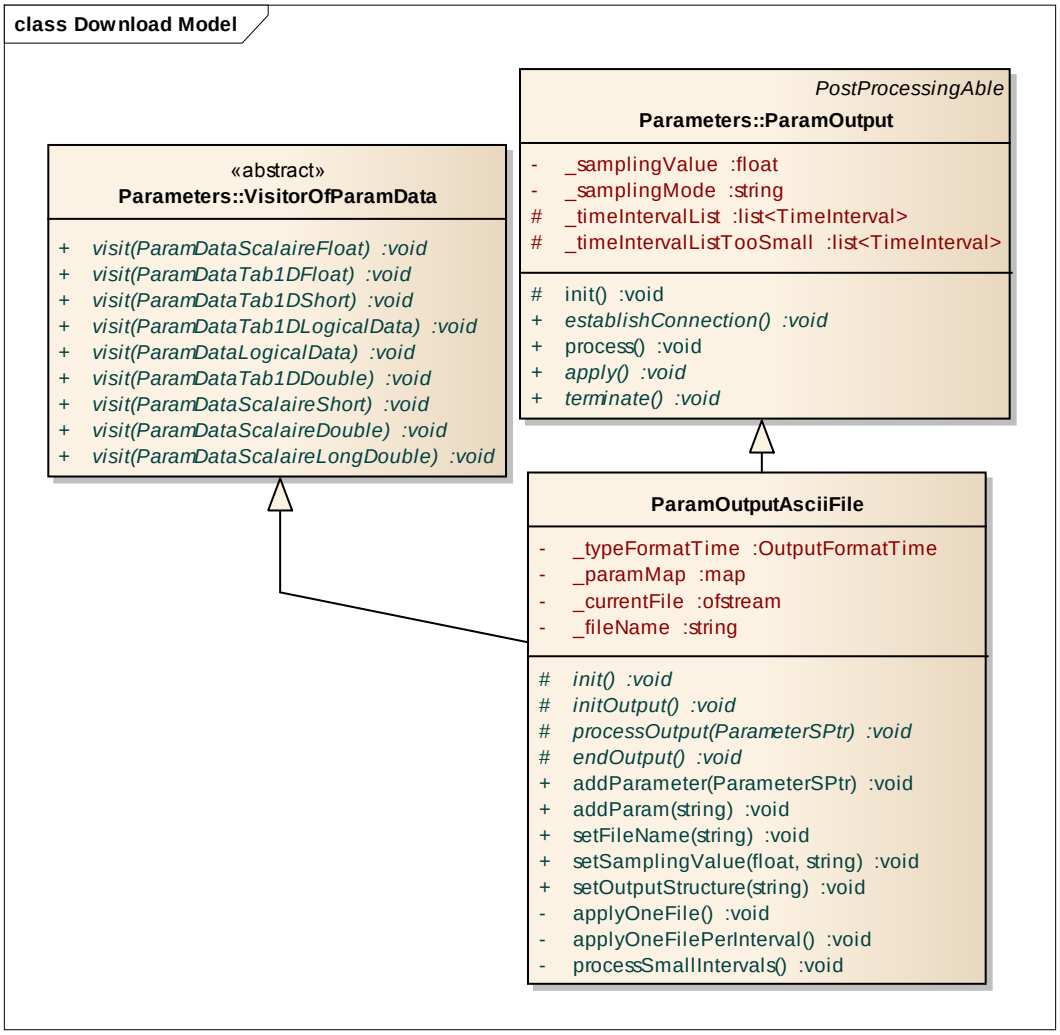


Figure 25 Classe ParamOutputAsciiFile

La classe responsable de la lecture du nœud `<param>` est modifiée en conséquence et de nouvelles classes pour la lecture des nœuds `<timeResolution>`, `<fileName>` et `<outputStructure>` sont ajoutées. La valeur de l'élément `<timeResolution>` est stockée dans l'attribut `_samplingValue` hérité de de la classe `ParamOutput`. Par ailleurs, `_samplingMode` est positionné par défaut à « `classic` » mais peut être modifié par le biais de la méthode `setSamplingValue(samplingValue, samplingMode)`.

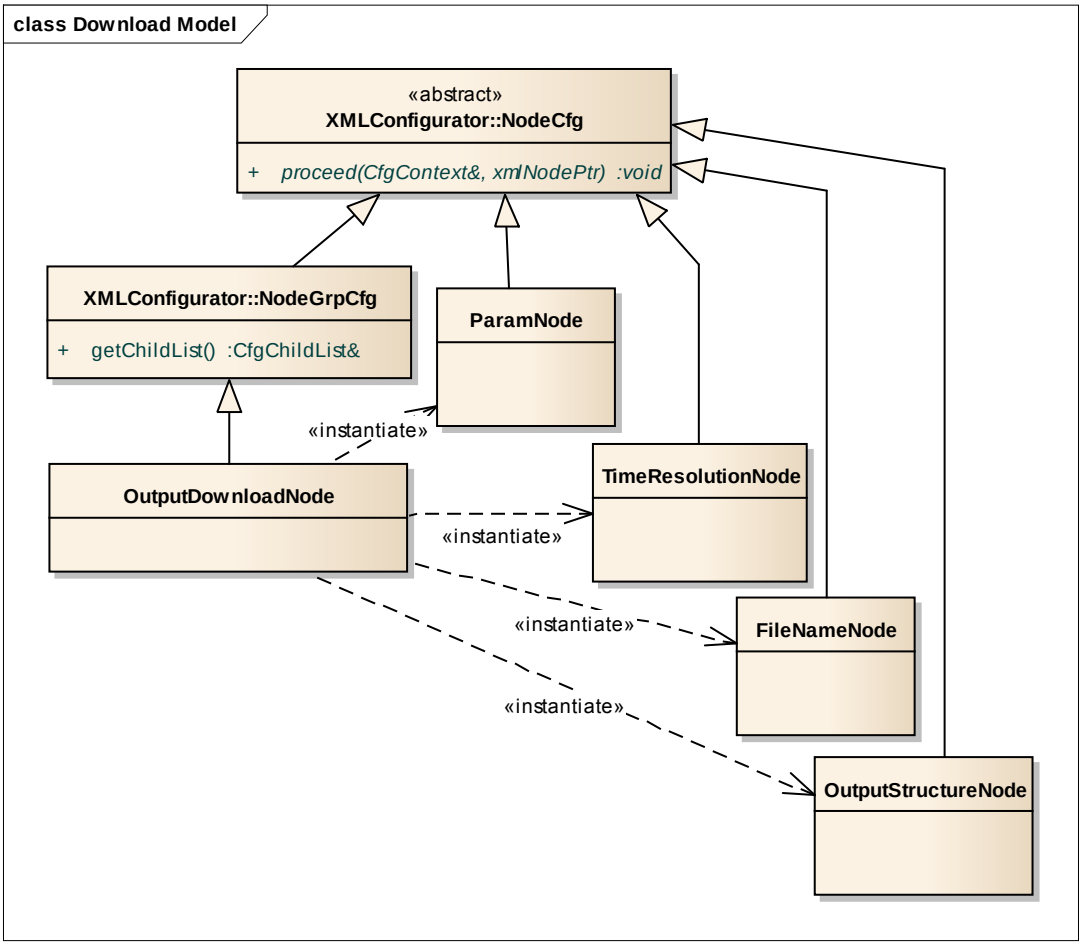


Figure 26 : Download & classes de lecture des noeuds XML

5.3.7.5 Dynamique

5.3.7.5.1 Traitement principal

Le traitement d'une requête de download sur plusieurs paramètres est illustré par le diagramme ci-après. Voici son déroulement :

1. Pour chacun des identifiants de paramètre de la requête, la méthode `getSampledParameter` crée le paramètre éventuellement ré échantillonné (le ré échantillonnage n'est effectué que si le mode et la valeur de ré-échantillonnage sont renseignés lors de l'appel à cette méthode) et ouvre une connexion sur ce paramètre.
2. Chacun des paramètres est ensuite initialisé.
3. Enfin, l'application du download (méthode `apply`) génère systématiquement un fichier par intervalle (méthode `applyOneFilePerInterval()`), ces derniers sont concaténés dans le cas d'un format de sortie `one-file` (méthode `applyOneFile()`). Le mécanisme de génération suit le processus suivant :
    a. Création et ouverture d'un fichier temporaire pour écrire l'ensemble des dates formatées correspondant à l'échantillonnage de la requête ou, à défaut, du premier paramètre dans ce fichier. Le buffer d'écriture est vidé à la fin de cette écriture.
    b. Demande de valeur pour le premier paramètre, parcours en lecture du fichier temporaire précédent et écriture dans un second fichier temporaire la ligne lue puis, pour la date correspondante, la valeur du second paramètre. Le buffer d'écriture est vidé à la fin de cette seconde écriture.

Remarque : Si le paramètre n'a pas de valeur pour une date donnée, la valeur complétée par l'instruction `<< NotANumber()` est inscrite.

c. Les autres éventuels paramètres sont traités comme le premier paramètre : le dernier fichier écrit est parcouru en lecture et chaque ligne est complétée dans un nouveau fichier.
d. Clôture du fichier et passage à l'intervalle suivant s'il existe (méthode `endOutput()`)

Le scénario ci-dessous présente le cas d'une requête de download pour une liste d'intervalles de temps de plusieurs paramètres dans un seul fichier.

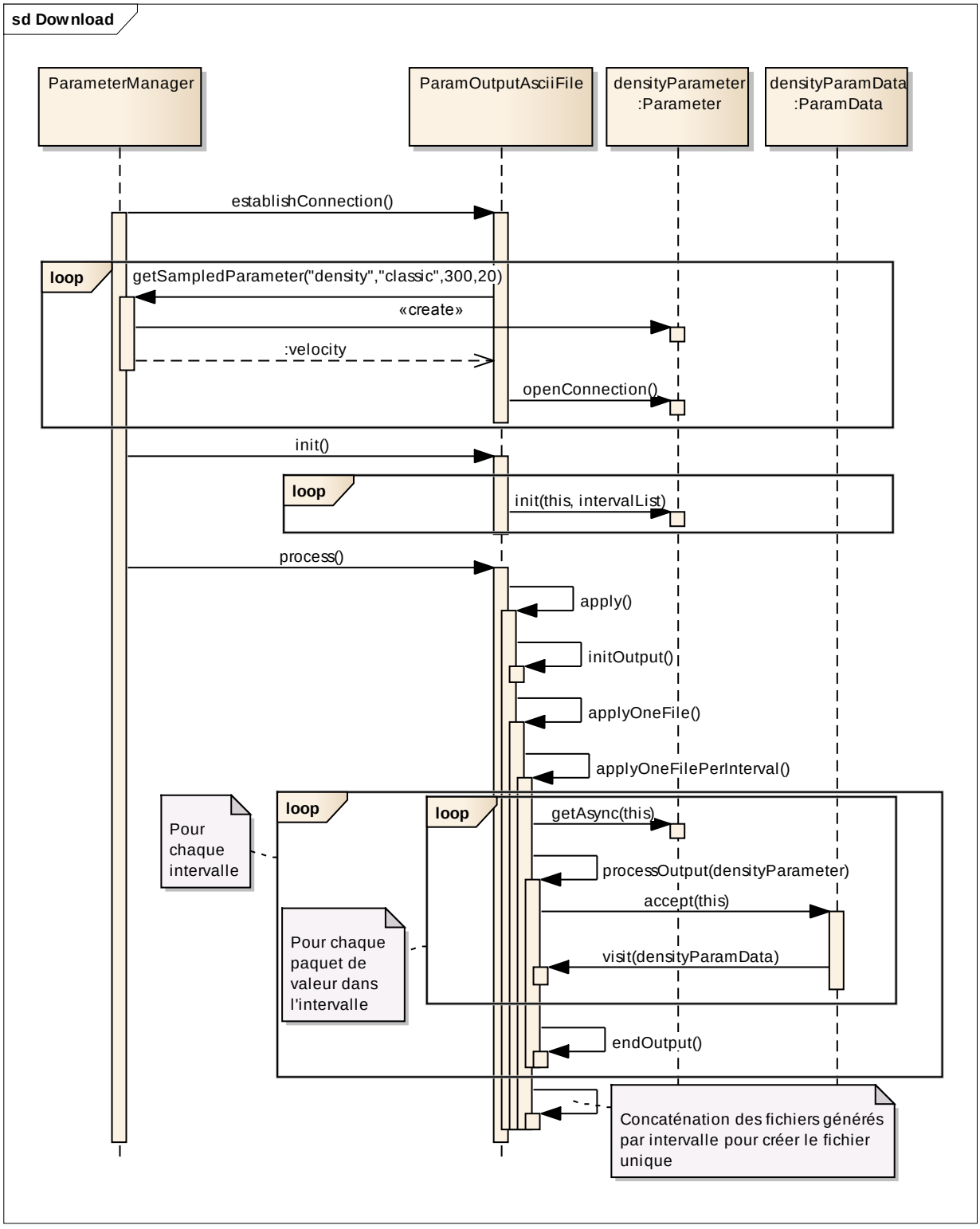


Figure 27 : Scénario Download de plusieurs paramètres

5.3.7.5.2 Traitement des intervalles trop petits

Le scénario ci-dessous présente le traitement des intervalles de temps trop petits (date de fin d'intervalle – date de début d'intervalle < résolution précisée dans la requête) dans le cadre d'une sortie `one-file-per-interval`. Ces derniers sont isolés lors de l'initialisation de la requête dans une liste dédiée

ParamOutput::_timeIntervalListTooSmall et traités après les autres intervalles via la méthode ParamOutputAsciiFile::processSmallIntervals().

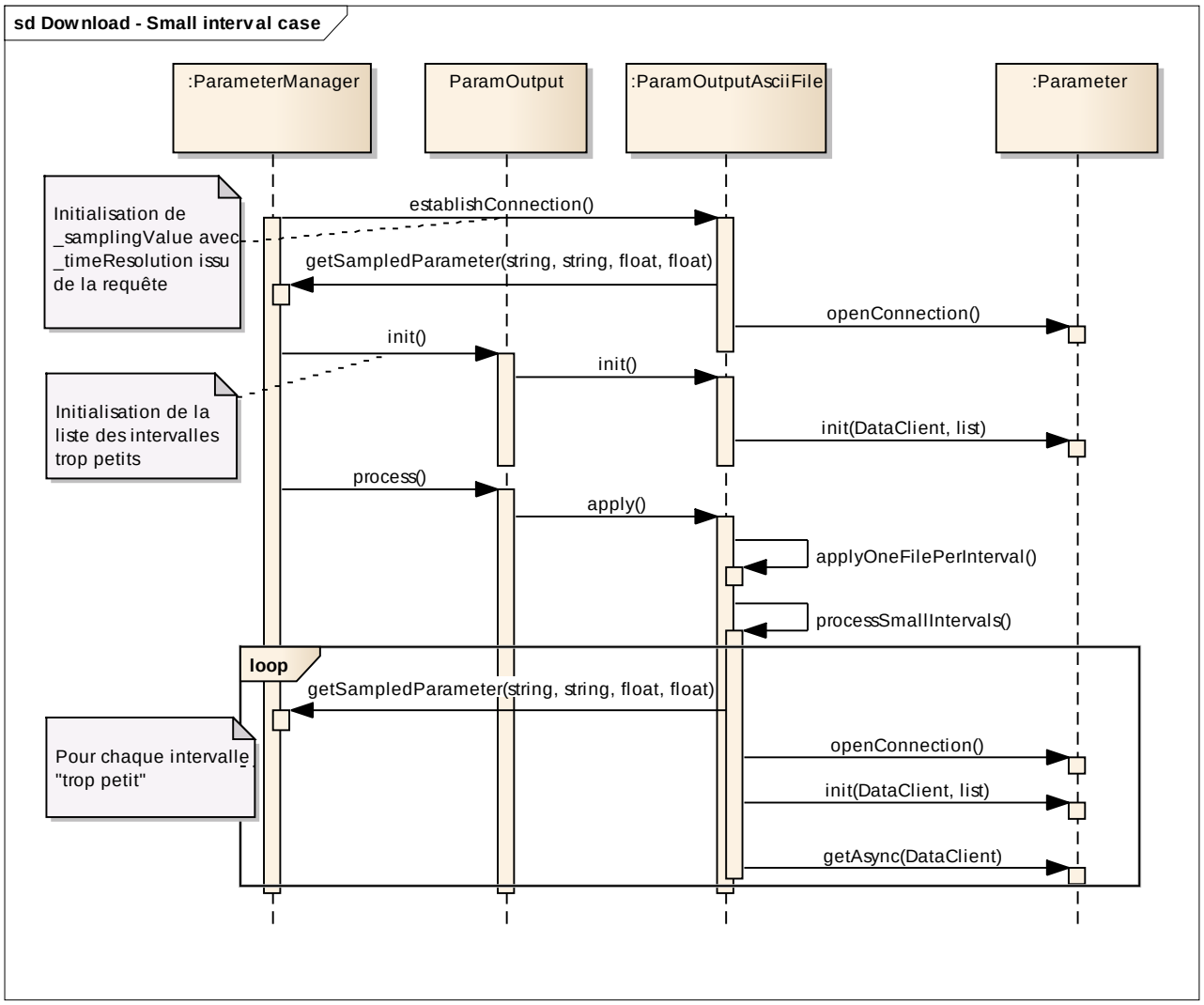


Figure 28 : Scénario Download - Traitement des intervalles trop petits

Lors du traitement des intervalles trop petits, le ou les paramètres sont ré-échantillonnés (méthode getSampledParameter()) selon l'intervalle de temps couramment traité (_startTime – _stopTime) et les valeurs sont inscrites dans un fichier dédié. Chaque intervalle de temps génère un fichier de sortie de nom spécifique indexé et précisant, en entête, l'information suivante :

```
# time resolution requested: XXX s
# time resolution used: XXX s
```

Dans le cadre d'une sortie one-file, la même méthode processSmallInterval() inscrit la liste des intervalles trop petits dans une timetable dédiée de nom <input-tt>_<timeResolution>_smallIntervals au format identique à la time table en entrée de la requête.

5.3.7.6 Dépendances

Le Composant Download exploite notamment les composants PostProcessing et TimeTable.

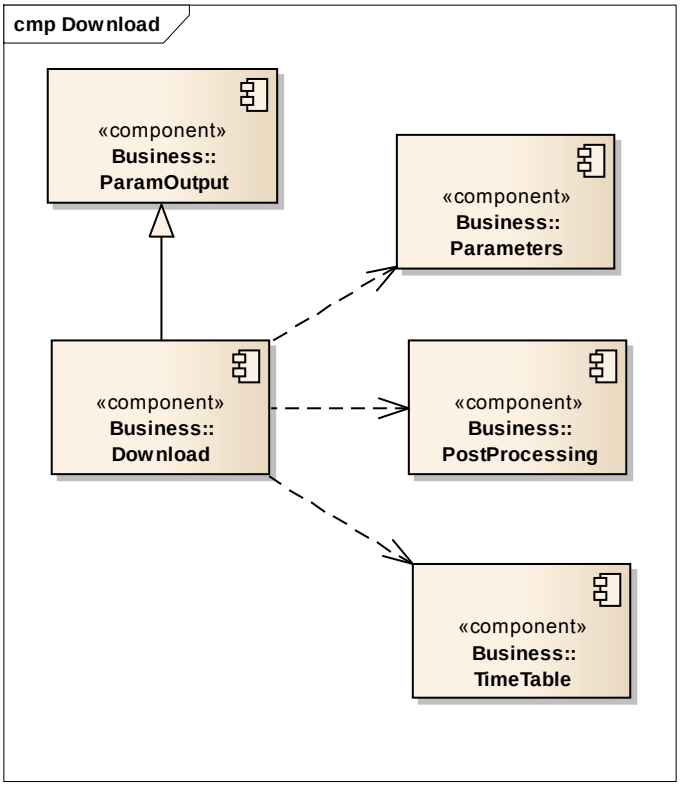


Figure 29 Dépendances du composant Download

5.3.8 Composant « Plot»

5.3.8.1 Exigences techniques

Id	Description	Commentaire
AMDA_PLOT_010	Le système permet, à partir d'une requête, de produire une visualisation de données portées par des paramètres.	STB §7, US14
AMDA_PLOT_020	Le système permet de positionner certaines valeurs par défaut ou imposées dans un fichier de configuration XML modifiable par un administrateur.	STB §7.1, US14
AMDA_PLOT_030	Le système permet d'organiser les éléments constituant la requête, du plus grand conteneur au plus petit, dans une 'page', un 'panel', un 'plot'.	STB §7.1.1, §7.1.2, US14

Tableau 7 : Exigences générales du composant Plot

5.3.8.1.1 Contexte

5.3.8.1.1.1 Page

Id	Description	Commentaire
AMDA_PLOT_040	Le format de sortie d'une 'page' est choisi parmi : png, pdf, ps et (éventuellement) svg	STB §7.1.1.1, US15

AMDA_PLOT_050	Les étapes techniques pour ajouter un nouveau format doivent être documentées	STB §7.1.1.1
AMDA_PLOT_060	La dimension d'une 'page' est définie soit: ✓ par la norme ISO A4 (210 mm * 297 mm) ✓ par le format US letter (215,9 mm * 279,4 mm) Par défaut, la page aura la dimension ISO A4.	STB §7.1.1.2, US15
AMDA_PLOT_070	Une 'page' peut être orientée en 'paysage' ou en 'portrait'. Par défaut, une 'page' est orientée en 'paysage'.	STB §7.1.1.3, US15
AMDA_PLOT_080	Une 'page' doit pouvoir se dessiner soit en mode 'couleur', soit en mode 'niveau de gris'. Par défaut, une 'page' se dessinera en mode 'couleur'.	STB §7.1.1.4, US15
AMDA_PLOT_090	Une police (type et taille) peut être définie au niveau de la 'page'. Cette police sera attribuée par défaut à un 'panel', sauf si une autre police lui est associée. Par défaut, la police de la page sera de type 'sans-serif' et sa taille de '12'.	STB §7.1.1.5, US15
AMDA_PLOT_100	Les marges représentent l'espacement laissé libre autour d'une page. Par défaut, les marges horizontales seront de 15 mm, et les marges verticales de 20 mm.	STB §7.1.1.6, US15
AMDA_PLOT_110	Dans une 'page', plusieurs 'panels' pourront être dessinés. Les 'panels' seront positionnés sur la page en suivant les règles imposées par le 'layout'.	STB §7.1.1.7, US15
AMDA_PLOT_120	L'information 'Created by AMDA', ainsi que la date et l'heure de la génération de la page, doivent être incluses en dessous du dernier 'panel' dessiné sur la 'page'.	STB §7.1.1.8, US15

Tableau 8 : Exigences du composant Plot - Page

5.3.8.1.1.2 Panel

Id	Description	Commentaire
AMDA_PLOT_130	La largeur et la hauteur d'un panel pourront être définies. Dans le cas où ces valeurs ne sont pas définies, la dimension des panels est déterminée par le 'layout'.	STB §7.1.2.1, US16
AMDA_PLOT_140	La position d'un 'panel' sur une 'page' pourra être définie. Dans ce cas, le positionnement du panel se fera en outrepassant toutes les règles définies par le layout.	STB §7.1.2.2, US16
AMDA_PLOT_150	Un 'panel' pourra contenir un titre, sur une ou plusieurs lignes. Il pourra être positionné en haut ou en bas du 'panel', et centré dans le sens de la largeur.	STB §7.1.2.3, US16
AMDA_PLOT_160	Une couleur de fond spécifique peut être définie, elle sera appliquée sur la zone de plot du panel. Par défaut, la couleur de fond est blanche.	STB §7.1.2.4, US16
AMDA_PLOT_170	Une police (type et taille) peut être définie au niveau d'un 'panel'. Par défaut, c'est la police de la 'page' qui sera utilisée.	STB §7.1.2.5, US16

AMDA_PLOT_180	Dans la majorité des cas, un 'panel' contiendra un ensemble d'axes, dont au moins un axe des abscisses et un axe des ordonnées. Un troisième type d'axe pourra être défini (correspondant à une 3ème dimension), permettant de représenter les données d'un paramètre sous forme de code couleurs. Seul le cas d'un plot d'orbites sous forme 'tickmarks' ne nécessitera pas la définition d'axes dans un panel.	STB §7.1.2.6, US17
AMDA_PLOT_190	Une résolution, exprimée en 'points par plot', pourra être définie (par défaut cette valeur vaudra 3000). Cela représente le nombre maximal de points à dessiner pour un paramètre sur l'intervalle de temps donné. Notons 'plot_res' la résolution d'un plot, 'delta_T' la durée de l'intervalle de temps de la requête, et 'param_res' la résolution temporelle (ou 'sampling time') du paramètre à ploter. Si 'param_res > plot_res/delta_T' alors le paramètre sera resamplé afin de le ramener à 'param_res' points sur l'intervalle de temps de la requête.	STB §7.1.2.7, US25
AMDA_PLOT_200	Une liste d'objets additionnels à dessiner sur un 'panel' peut être fournie.	STB §7.1.2.8

Tableau 9 : Exigences du composant Plot -Panel

5.3.8.1.1.3 Layout

Id	Description	Commentaire
AMDA_PLOT_210	Par défaut, les 'panels' sont positionnés les uns en dessous des autres dans une 'page', sans espacement.	STB §7.1.3
AMDA_PLOT_220	La définition d'un 'layout' permet à un utilisateur de positionner des 'panels' sur une 'page', en définissant un ensemble de règles. Lorsqu'un 'panel' contient des informations de positionnement, le 'layout' n'aura aucun effet sur celui-ci.	STB §7.1.3
AMDA_PLOT_230	Un espacement vertical et un espacement horizontal entre deux 'panels' contigus pourront être définis. Par défaut, ces valeurs vaudront 0 mm.	STB §7.1.3.1
AMDA_PLOT_240	Dans le cas où un 'panel' ne contient pas d'information de dimension, c'est au 'layout' que revient la responsabilité de le dimensionner, en respectant les règles suivantes : ✓ les 'panels' représentant une série temporelle (ie. l'axe des abscisses porte un temps) prendront toute la largeur de la page ✓ les 'panels' représentant un scatter plot (ie. l'axe des abscisses porte un paramètre) prendront pour largeur la hauteur du panel sans les 'objets additionnels' (cf. section 'Objets Additionnels') la hauteur par défaut d'un 'panel'	STB §7.1.3.2
AMDA_PLOT_250	Lorsque l'auto layout est activé, le système cherchera à optimiser le remplissage d'une 'page' en positionnant plusieurs 'panels' contigus sur la même ligne (lorsque cela est possible). Par défaut, l'auto layout est désactivé.	STB §7.1.3.3

Tableau 10 : Exigences du composant Plot - Layout

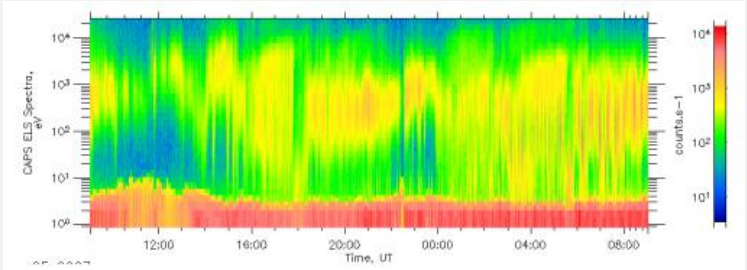
5.3.8.1.1.4 Axe

Id	Description	Commentaire
AMDA_PLOT_260	Il existe quatre types d'axes: ✓ l'axe x : axe des abscisses portant des données de paramètres ✓ l'axe y : axe des ordonnées portant des données de paramètres ✓ l'axe t : axe des abscisses portant un temps ✓ l'axe epoch t : axe des abscisses portant un temps relatif au temps de référence '0 Une combinaison 'axe t / axe y' sera utilisée dans le cas d'un plot d'une série temporelle (time-serie), une combinaison 'axe x / axe y' sera utilisée dans le cas d'un scatter plot, et une combinaison 'axe epoch t / axe y' sera utilisée dans le cas d'un plot de type 'superposed epoch analysis' (cf. section 'Types de plot'). A noter qu'un 'panel' ne peut avoir qu'un seul 'axe t' ou 'axe epoch t', alors qu'il peut avoir plusieurs 'axe x' ou 'axe y'.	STB §7.1.4.1, US17
AMDA_PLOT_270	Les bornes d'un axe correspondent à une 'valeur minimale' et à une 'valeur maximale' de définition. Les bornes d'un 'axe t' seront obligatoirement le temps de départ et le temps de fin de la requête (imposé par le système). Les bornes d'un 'axe x' ou d'un 'axe y' sont automatiquement attribuées par le système lorsque l'auto scale est activé. Les bornes d'un 'axe epoch t' ne seront pas indiquées, elles découleront de la définition du 'delta t inf.' et du 'delta t sup.'	STB §7.1.4.2, US17
AMDA_PLOT_280	'Delta t inf.' et 'Delta t sup.' définiront les bornes d'un 'axe epoch t' par rapport au temps de référence '0'.	STB §7.1.4.3, US17
AMDA_PLOT_290	Lorsque l'auto scale est activé sur un axe, les bornes de celui-ci seront automatiquement attribuées en fonction des paramètres qu'il porte, de manière à ce que l'ensemble des données soient visibles sur le plot. L'auto scale n'a pas d'effet sur un sur un 'axe t' ou un 'axe epoch t'.	STB §7.1.4.4, US17
AMDA_PLOT_300	Par défaut, l'origine d'un axe correspond à sa borne minimale. L'utilisateur pourra définir sa propre valeur d'origine. Si l'origine définie est en dehors des bornes de l'axe, elle sera remplacée par la borne minimale. Dans le cas d'un 'axe epoch t', l'origine sera systématiquement placée au temps de référence '0'.	STB §7.1.4.5, US17
AMDA_PLOT_310	Un axe sera accompagné d'une graduation et de valeurs étiquetées afin de pouvoir se repérer visuellement sur celui-ci. La graduation s'adaptera en fonction des valeurs de ses bornes, afin que l'information reste toujours lisible.	STB §7.1.4.6, US17

AMDA_PLOT_320	De plus, dans le cas d'un 'axe t', le formatage du temps pourra être défini: ✓ 'jour julien' ✓ 'jour/moi/année' ✓ 'numjour/année' Par défaut, c'est le format 'jour/moi/année' qui sera utilisé.	STB §7.1.4.6, US17
AMDA_PLOT_330	L'échelle d'un 'axe x' ou d'un 'axe y' peut être linéaire ou logarithmique. Par défaut, un axe est linéaire.	STB §7.1.4.7, US17
AMDA_PLOT_340	Par défaut: ✓ les 'axe x', 'axe t' et 'axe epoch t' sont orientés de gauche à droite ✓ les 'axe y' sont orientés de bas en haut L'orientation d'un axe peut être inversée en activant cette option.	STB §7.1.4.8, US17
AMDA_PLOT_350	Une légende doit accompagner chaque axe pour le décrire. Une légende peut s'écrire sur plusieurs lignes.	STB §7.1.4.9, US17
AMDA_PLOT_360	Une couleur spécifique peut être rattachée à un axe. La couleur par défaut d'un axe est noir. Pour une série temporelle, les 'paramètres' rattachés à un 'axe y' adopteront par défaut sa couleur.	STB §7.1.4.10, US17
AMDA_PLOT_370	Un axe x pourra être positionné en haut ou en bas de la zone à plotter (par défaut en bas). Un axe y pourra être positionné à gauche ou à droite de la zone à plotter (par défaut à gauche). Un 'axe t' ou un 'axe epoch t' sera positionné en bas de la zone à plotter (imposé par le système).	STB §7.1.4.11, US17
AMDA_PLOT_380	Plusieurs 'axe x' et 'axe y' peuvent être définis sur un 'panel'.	STB §7.1.4.12, US17

Tableau 11 : Exigences du composant Plot - Axe

5.3.8.1.1.5 Color Axe

Id	Description	Commentaire
AMDA_PLOT_390	L'objet 'colour axe' permet de représenter sur un plot une troisième dimension, en faisant apparaître les valeurs d'un paramètre sous forme d'un code couleur. 	STB §7.1.5
AMDA_PLOT_400	Les bornes d'un 'colour axe' correspondent à une 'valeur minimale' et à une 'valeur maximale' de définition. Les bornes d'un axe sont automatiquement attribuées par le système lorsque l'auto scale est activé.	STB §7.1.5.1

AMDA_PLOT_410	Lorsque l'auto scale est activé sur un 'colour axe', les bornes de celui-ci seront automatiquement attribuées en fonction de la valeur minimale et de la valeur maximale du paramètre qu'il porte.	STB §7.1.5.2
AMDA_PLOT_420	Une table de couleur à utiliser pourra être définie: ✓ 'arc-en-ciel'  ✓ 'niveaux de gris'  ✓ 'bicolore rouge-bleu'  Par défaut, la table de couleurs sera 'arc-en-ciel' si la 'page' est en mode 'couleur' et 'niveaux de gris' si la 'page' est en 'niveau de gris'.	STB §7.1.5.3
AMDA_PLOT_430	La valeur moyenne associée à la couleur centrale de la table de couleur pourra être définie. Par défaut, cette valeur correspond au centre des bornes du 'colour axe'.	STB §7.1.5.4
AMDA_PLOT_440	L'échelle d'un 'Colour Axe' peut être linéaire ou logarithmique. Par défaut, elle sera linéaire.	STB §7.1.5.5
AMDA_PLOT_450	L'orientation d'un 'colour axe' peut être inversée en activant cette option.	STB §7.1.5.6
AMDA_PLOT_460	Une légende doit accompagner chaque 'colour axe' pour le décrire.	STB §7.1.5.7

Tableau 12 : Exigence du composant Plot - Color Axe

5.3.8.1.1.6 Objets additionnels

Id	Description	Commentaire
AMDA_PLOT_470	Différents objets additionnels peuvent être dessinés sur un 'panel' afin d'y apporter des informations supplémentaires.	STB §7.1.6
AMDA_PLOT_480	Il doit être possible de rajouter d'autres types d'objets additionnels et les étapes pour le faire doivent être documentées.	STB §7.1.6.1
AMDA_PLOT_490	'Constante' Une ou plusieurs constantes peuvent être associées à un axe, et seront tracées sous forme de lignes verticales ou horizontales sur la zone de plot, en fonction du type de l'axe. La couleur et l'épaisseur de la ligne pourra être définie.	STB §7.1.6.1
AMDA_PLOT_500	'Intervalle' Un ou plusieurs intervalles de temps associés à un 'axe t' pourront être tracés sur la zone de plot sous forme de blocs translucides.	STB §7.1.6.1
AMDA_PLOT_510	'Label' Des 'labels' (textes descriptifs) pourront être insérés dans la zone de plot d'un panel. La police utilisée sera celle du 'panel'. La couleur du 'label' pourra être définie.	STB §7.1.6.1

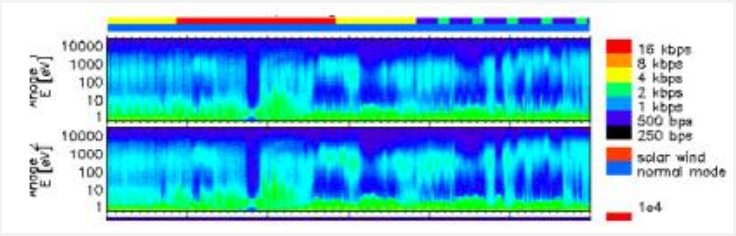
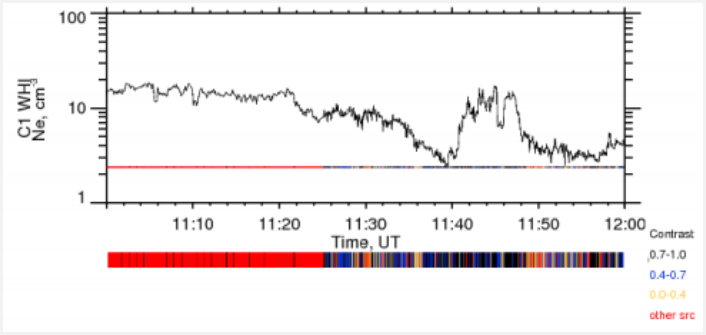
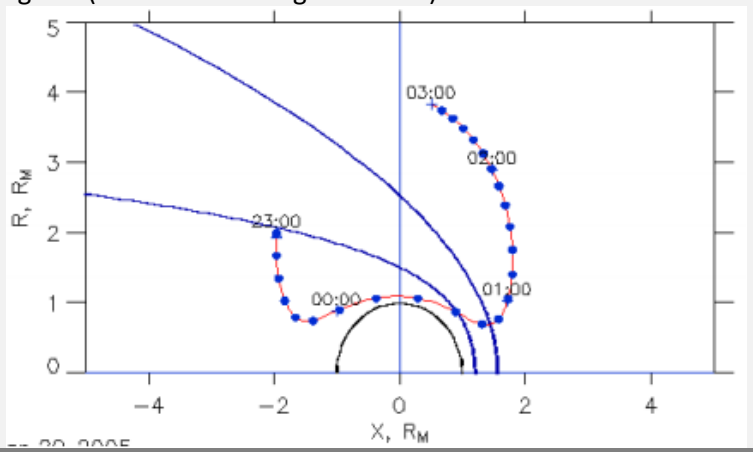
AMDA_PLOT_520	'Colour Band' Une ou plusieurs 'colour bands' peuvent être associés à un 'panel' et obligatoirement reliées à un 'axe t' (=séries temporelles). Une 'colour band' permettra d'indiquer un 'état' (par exemple un mode de télémétrie) par le biais d'un code couleur.   Une 'colour band' pourra être positionnée en haut ou en bas d'un 'panel'. Une 'color band' se dessinant en dessous ou au-dessus de la zone de plot, sa hauteur devra être définie (elle sera utilisée par le 'layout' pour déterminer la hauteur total du 'panel', sauf si la hauteur du 'panel' est imposée).	STB §7.1.6.1
AMDA_PLOT_530	'Représentation d'une planète ou d'une région modélisée' Des informations supplémentaires pourront être ajoutées sur la zone de plot, et notamment pour les plots d'orbites, comme la représentation de la planète cible, et des modélisations de régions (en bleu sur l'image suivante).  	STB §7.1.6.1

Tableau 13 : Exigences du composant Plot - Objets additionnels

5.3.8.1.2 Plot de paramètres

Id	Description	Commentaire
AMDA_PLOT_540	Un plot est systématiquement associé à un panel.	STB §7.2.1, US16
AMDA_PLOT_550	Le type de plot doit être défini et sera choisi parmi : ✓ série temporelle ✓ série XY (scatter plot) ✓ spectrogramme série temporelle ✓ matrice : représentation d'une matrice pour un temps donné ✓ statut : utilisation d'une 'colour band' pour montrer un statut ✓ série temporelle avec 'quality flag' : il s'agit d'une série temporelle dont la qualité des données est représentée par un code couleur sur la courbe. ✓ orbite tickmarks : les informations d'orbites pourront être représentées sous forme de tickmarks: ✓ superposed epoch analysis : une liste d'événements temporelles sera superposée sur un même plot (l'axe des abscisses sera alors un axe de temps relatifs à ces événements) ✓ position orbitale : la position orbitale d'un ou de plusieurs satellites pour un temps donné	STB §7.2.2, US18, US19, US20

Tableau 14 : Exigences du composant Plot – Paramètres

Exemples de types de plots :

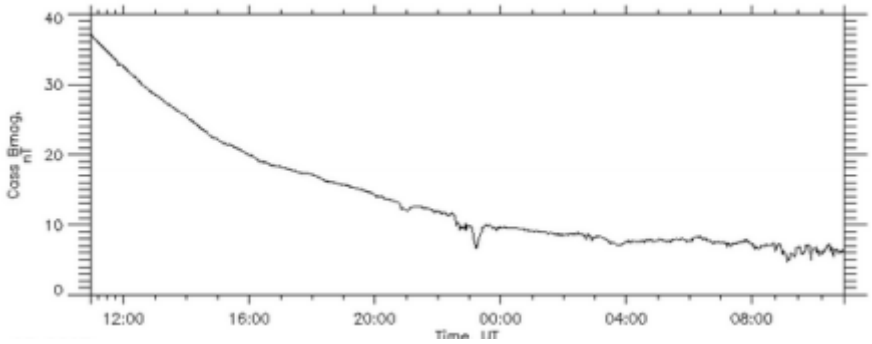


Figure 30 : Exemple de plot - Série temporelle

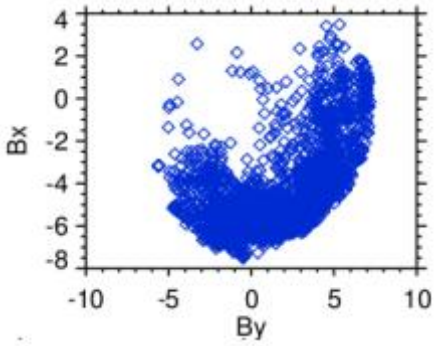


Figure 31 : Exemple de plot - Scatter plot

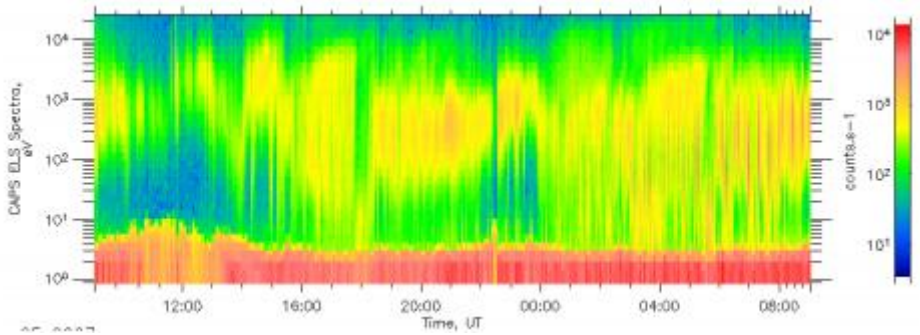


Figure 32 : Exemple de plot – Spectrogramme

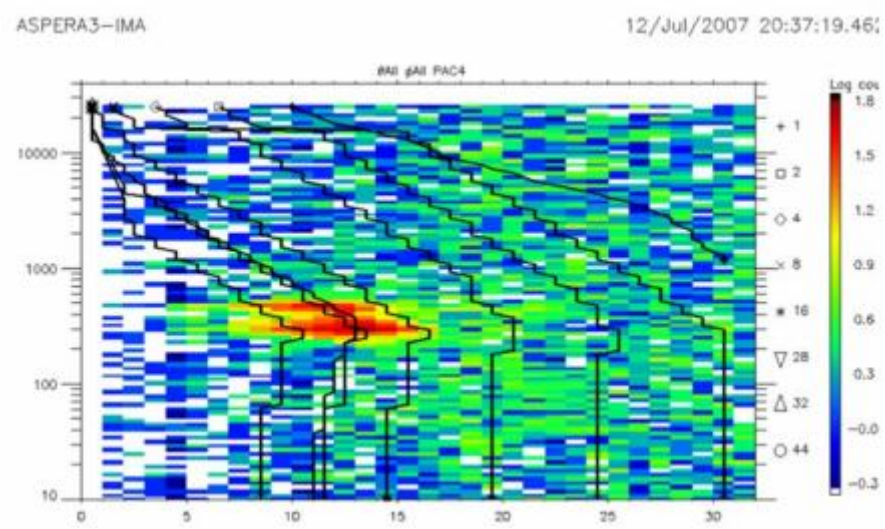


Figure 33 : Exemple de plot - Matrice

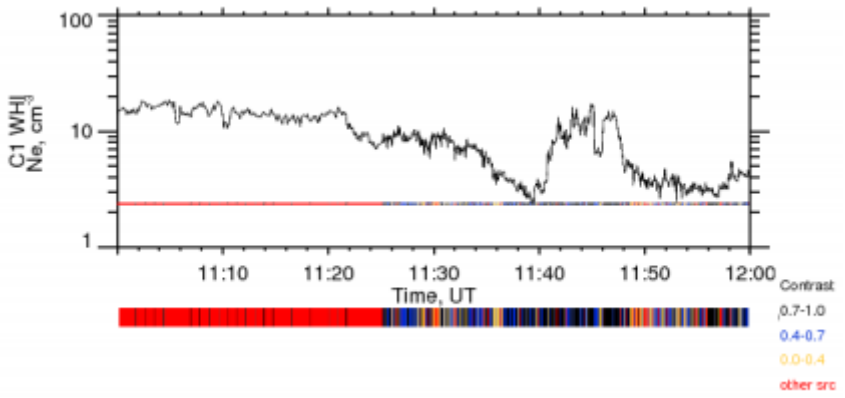


Figure 34 : Exemple de plot - Statut

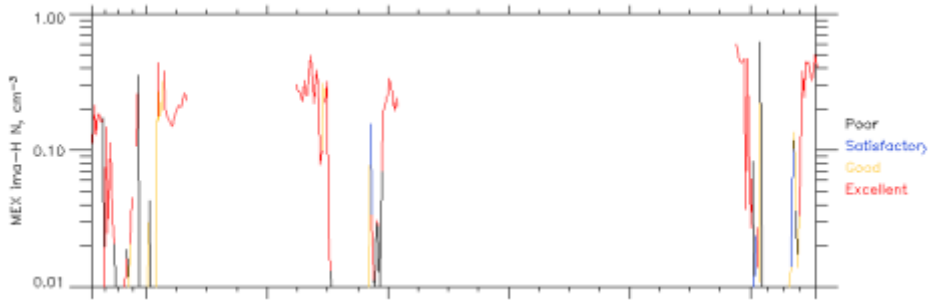


Figure 35 : Exemple de plot - Série temporelle avec 'Quality Flag'

Time, UT	16:00	20:00	00:00	04:00	08:00	12:00
MGS R PC	1.121	1.121	1.122	1.122	1.122	1.121
Lon PC	19.231	−41.220	−105.290	146.142	−16.991	−83.036
Lat PC	47.785	61.007	74.403	86.606	78.010	64.697

Figure 36 : Exemple de plot - Orbit Tickmarks

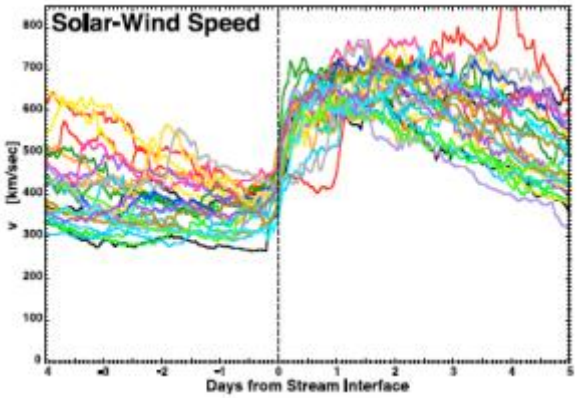


Figure 37 : Exemple de plot - Superposed epoch analysis

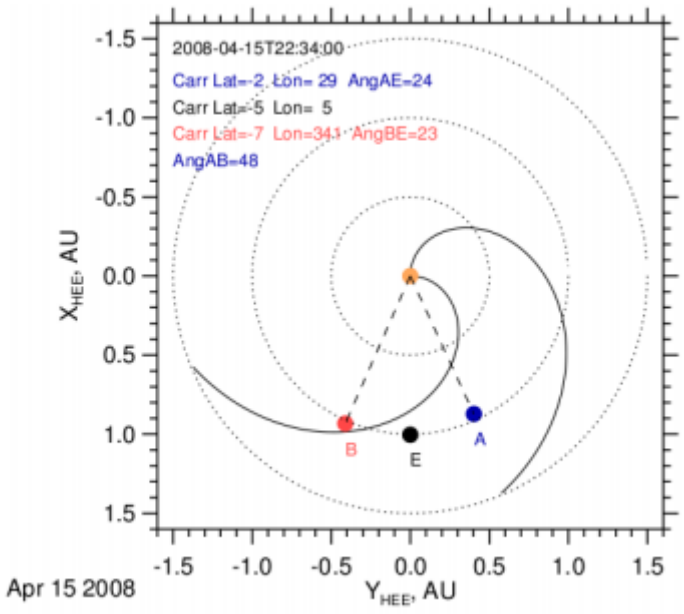
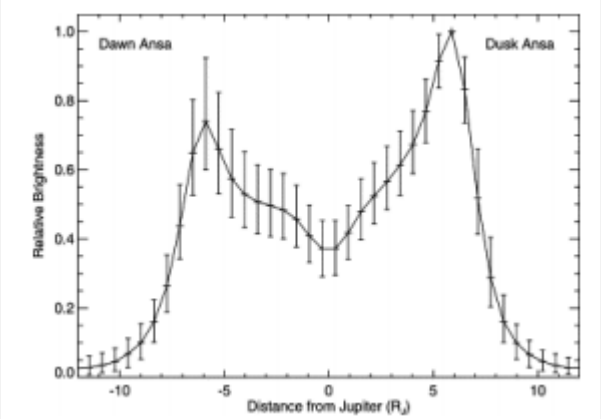


Figure 38 : Exemple de plot - Position orbitale

Id	Description	Commentaire
AMDA_PLOT_560	Il doit être possible d'ajouter un nouveau type de plot et les étapes pour le faire doivent être documentées.	STB §7.2.2
AMDA_PLOT_570	En fonction du type de plot, un certain nombre d'axes du panel courant seront utilisés : ✓ pour une série temporelle : ➢ un 'axe y' (requis) ➢ l'axe t du panel courant sera automatiquement utilisé (imposé par le système) ➢ un 'colour axe' peut être défini (optionnel) pour une série XY (scatter plot) : ➢ un 'axe x' (requis) ➢ un 'axe y' (requis) ➢ un 'colour axe' peut être défini (optionnel) ✓ pour un spectrogramme : ➢ un 'axe y' (requis) ➢ l'axe t du panel courant sera automatiquement utilisé (imposé par le système) ➢ un 'colour axe' (requis) ✓ pour une matrice : ➢ un 'axe x' (requis) ➢ un 'axe y' (requis) ➢ un 'colour axe' (requis) ✓ pour un statut : ➢ l'axe t du panel courant sera automatiquement utilisé (imposé par le système) ➢ la 'colour band' à utiliser devra être indiquée (requis) ➢ pour une série temporelle avec 'quality flag' : ➢ l'axe t du panel courant sera automatiquement utilisé (imposé par le système) ➢ un 'axe y' (requis) ✓ pour une 'orbite tickmarks' : aucun axe n'est requis ✓ pour un superposed epoch analysis : ➢ un 'axe y' (requis) ➢ l'axe epoch t du panel courant sera automatiquement utilisé (imposé par le système)	STB §7.2.3, US18, US19, US20
AMDA_PLOT_580	✓ pour une série temporelle, les paramètres AMDA suivants seront à indiquer : ➢ un paramètre ou une composante d'un paramètre (requis), de type scalaire ou 'tenseur d'ordre 1' : ➢ un paramètre ou une composante d'un paramètre (optionnel), de type scalaire à utiliser avec le 'colour axe' :	STB §7.2.4, US18

AMDA_PLOT_590	✓ pour une série XY (scatter plot), les paramètres AMDA suivants seront à indiquer : ➢ un paramètre ou une composante d'un paramètre (requis), de type scalaire porté par l'axe x ➢ un paramètre ou une composante d'un paramètre (requis), de type scalaire porté par l'axe y ➢ un paramètre ou une composante d'un paramètre (optionnel), de type scalaire à utiliser avec le 'colour axe' :	STB §7.2.4, US19
AMDA_PLOT_600	✓ pour un spectrogramme, les paramètres AMDA suivants seront à indiquer : ➢ une table d'énergies ou de fréquences (requis) porté par l'axe y (peut être dépendant du temps) ➢ un paramètre (requis), de type 'tenseur d'ordre 2', dont une dimension représente le temps (porté par l'axe t), et l'autre l'index dans la table d'énergie (donc à ramener sur l'axe y). Les données du paramètres seront quant à elles portées par le 'colour axe'.	STB §7.2.4
AMDA_PLOT_610	✓ pour une matrice, les paramètres AMDA suivants seront à indiquer : ➢ le temps t de référence ➢ une table portée par l'axe x (requis) ➢ une autre table portée par l'axe y (requis) ➢ un paramètre (requis), de type 'tenseur d'ordre 2', dont une dimension représente l'index dans la table portée par l'axe x, et l'autre l'index dans la table portée par l'axe y. Les données du paramètres seront quant à elles portées par le 'colour axe'.	STB §7.2.4
AMDA_PLOT_620	✓ pour un statut, les paramètres AMDA suivants seront à indiquer : ➢ un paramètre ou une composante d'un paramètre (requis), de type scalaire entier, porté par la 'colour band' ➢ ou une 'time table' (liste d'intervalles). Dans ce cas-là : ▪ le statut sera 1 lorsque un temps est inclus par la 'time table' ▪ le statut sera 0 dans le cas contraire ▪ ce statut sera représenté sur la 'colour band'	STB §7.2.4
AMDA_PLOT_630	✓ pour une série temporelle avec 'quality flag', les paramètres AMDA suivants seront à indiquer : ➢ un paramètre ou une composante d'un paramètre (requis), de type scalaire ➢ un paramètre représentant le 'quality flag' (requis), de type scalaire ➢ une liste d'intervalles faisant le lien entre les données du 'quality flag' et l'appréciation de la qualité (poor, satisfactory, good, excellent)	STB §7.2.4

AMDA_PLOT_640	✓ pour une 'orbite tickmarks', les paramètres AMDA suivants seront à indiquer : ➢ un paramètre ou une composante d'un paramètre représentant une orbite de type scalaire ou tenseur d'ordre 1	STB §7.2.4, US20
AMDA_PLOT_650	✓ pour un superposed epoch analysis, les paramètres AMDA suivants seront à indiquer : ➢ un paramètre ou une composante d'un paramètre (requis), de type scalaire porté par l'axe y ➢ une liste d'événements temporels qui seront utilisés comme temps de référence	STB §7.2.4
AMDA_PLOT_660	Pour un plot d'une série temporelle ou d'un 'scatter plot', un autre paramètre (ou une composante d'un paramètre), de type scalaire, représentant l'erreur pourra être associé à un paramètre porté par l''axe x' ou l''axe y'. Il sera représenté sous forme de 'barres d'erreurs'. 	STB §7.2.5
AMDA_PLOT_670	Des options supplémentaires dédiées au cas d'un plot de type 'superposed epoch analysis' sont demandées: ✓ valeurs moyennes sur les différents événements (dans l'exemple suivant trois plots de trois paramètres différents sont dessinés sur un même panel) ✓ limites hautes et basses de la statistique (ex., quartiles, 10 et 90 percentile ou autres) sur les différents événements, qui pourront être représentées sous forme: ➢ de courbes (la zone séparant ces deux courbes pourra être coloriée) : ➢ de barres d'erreurs :	STB §7.2.6
AMDA_PLOT_680	Les isocontours sont des lignes d'isovaleurs dans un plot de type spectrogramme. Au lieu de remplir continument le plot avec des couleurs, on discrétise avec des valeurs prédéfinies (et souvent equi-espacées) données en option du plot.	STB §7.2.7
AMDA_PLOT_690	Par défaut, un paramètre d'une série temporelle portera la couleur de l'axe auquel il est associé (sauf si un 'colour axe' lui est associé). Dans le cas où le paramètre est de type 'tenseur d'ordre 1', une couleur spécifique sera automatiquement allouée par le système pour chaque composantes (sauf si une couleur est définie au niveau des options de plot):	STB §7.2.8, US18, US19

AMDA_PLOT_700	Le symbole représente la manière dont un point de donnée sera représenté. Les options disponibles seront : ✓ type de symbole : à choisir parmi une liste prédéfinie (aucun, point, plus (+), astérisque (*), diamant, triangle, carré, croix, etc...) Par défaut : 'aucun' si une ligne est définie, 'point' sinon ✓ taille de symbole Par défaut : 4 mm ✓ couleur Par défaut : pour une série temporelle, la couleur associé au paramètre ; pour un 'scatter plot', bleu	STB §7.2.9.1, US18, US19
AMDA_PLOT_710	La ligne représente la manière avec laquelle deux points de données consécutifs seront reliés entre eux. Les options disponibles seront : ✓ type de ligne : à choisir parmi une liste prédéfinie (aucun, linéaire, histogramme Par défaut 'aucun' pour un scatter plot, 'linéaire' pour une série temporelle ✓ forme de ligne : à choisir parmi 'pleine', ou différents types de 'pointillés Par défaut : pleine ✓ épaisseur de ligne Par défaut : 1 mm ✓ couleur Par défaut : pour une série temporelle, la couleur associée au paramètre ; pour un 'scatter plot', Rouge	STB §7.2.9.2, US18, US19
AMDA_PLOT_720	Dans le cas d'une série temporelle, des 'remplissages' d'une zone pourra être demandé: ✓ remplissage de la zone entre le paramètre porté par l'axe y (de type scalaire) et une constante temporelle : ➢ la constante en question pourra être définie au niveau des 'objets additionnels' du contexte du plot pour être dessinée sur le panel courant, ou non ➢ une option devra indiquer si c'est la zone au-dessus, ou en-dessous, de la constante qui doit être remplie ➢ une autre option indiquera la couleur à appliquer ✓ remplissage de la zone entre le paramètre porté par l'axe y (de type scalaire) et un autre paramètre (de type scalaire lui aussi)	STB §7.2.9.3

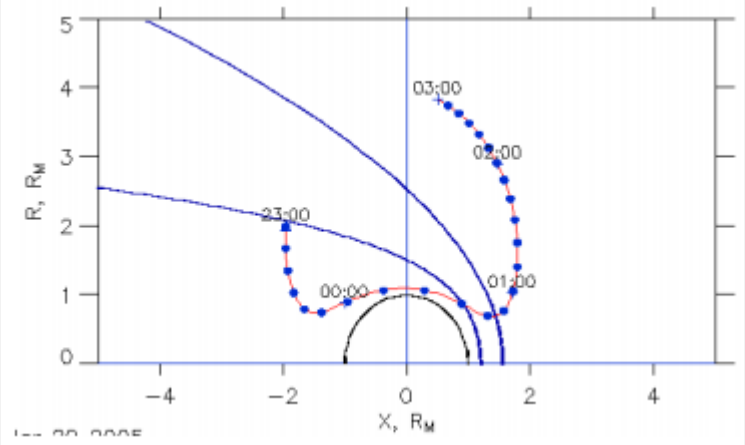
AMDA_PLOT_730	Des indications temporelles pourront être ajoutées sur un 'scatter plot'. 	STB §7.2.9.4

Tableau 15 : Exigences du composant Plot – Paramètres (suite)

5.3.8.2 Objet

Le composant Plot permet de produire une visualisation des données portées par des paramètres sous différentes représentations. Chaque type de visualisation est défini par une requête particulière et réalisé par un type d'output spécifique.
Les types de visualisation suivants sont définis :

- ✓ **TimePlot** : série temporelle
- ✓ **XYPlot :** série XY (scatter plot)
- ✓ **TickPlot** : représentation des valeurs d'un ou plusieurs paramètre sous la forme de « tickmarks ».

Bien que ces types de visualisation soient différents, ils partagent un ensemble d'informations similaires et le modèle objet permettant de les manipuler permet de mutualiser la plupart des notions.
Pour tous ces types de sorties, deux éléments sont essentiels :

- ✓ le contexte de tracé : il définit l'environnement du graphique (les propriétés de la page, du panel etc.).
- ✓ les paramètres à tracer : ils représentent les séries qui seront tracées.

Une requête peut contenir la demande de visualisation de plusieurs types de tracés différents dans une même page. Afin de gérer cette possibilité, il est nécessaire d'introduire un type de sortie général qui sera associé à la page et qui délèguera le tracé de chaque graphe à un output particulier. Ainsi, une requête de « Plot » se traduira au niveau du noyau AMDA-NG par la création d'un `PlotOutput` contenant des outputs spécialisés pour chaque graphe souhaité.

5.3.8.3 Classe PlotOutput et types de visualiation

Le module Plot est un plugin au sens AMDA-NG, à l'instar des autres types d'output. Chaque type de visualisation (timeplot, xyplot etc.) est également un plugin.

5.3.8.3.1 Classe PlotOutput : un manager

Une requête de Plot est traitée par la classe `PlotOutput` qui implémente `ParamOutput`. Celle-ci contient une liste de `PanelPlotOutput`, ces derniers se comportent également comme des `ParamOutput` mais contrairement aux autres type d'output (download, datamining …), ils ne sont pas

gérés directement par le `ParameterManager` mais par `PlotOutput` qui joue alors un rôle de « manager ». Il existe une implémentation de `PanelPlotOutput` pour chaque type de visualisation.

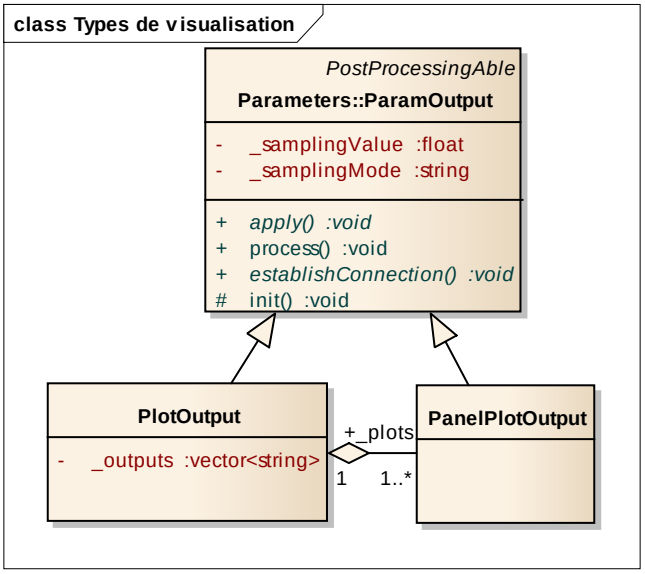


Figure 39 : PlotOutput et PanelPlotOutput

La séquence d'appel `establishConnection()`, `init()`, `process()` est relayée séquentiellement par `PlotOutput` vers chacun des `PanelPlotOutput`. `PlotOutput` est également responsable du dessin de la `Page`.

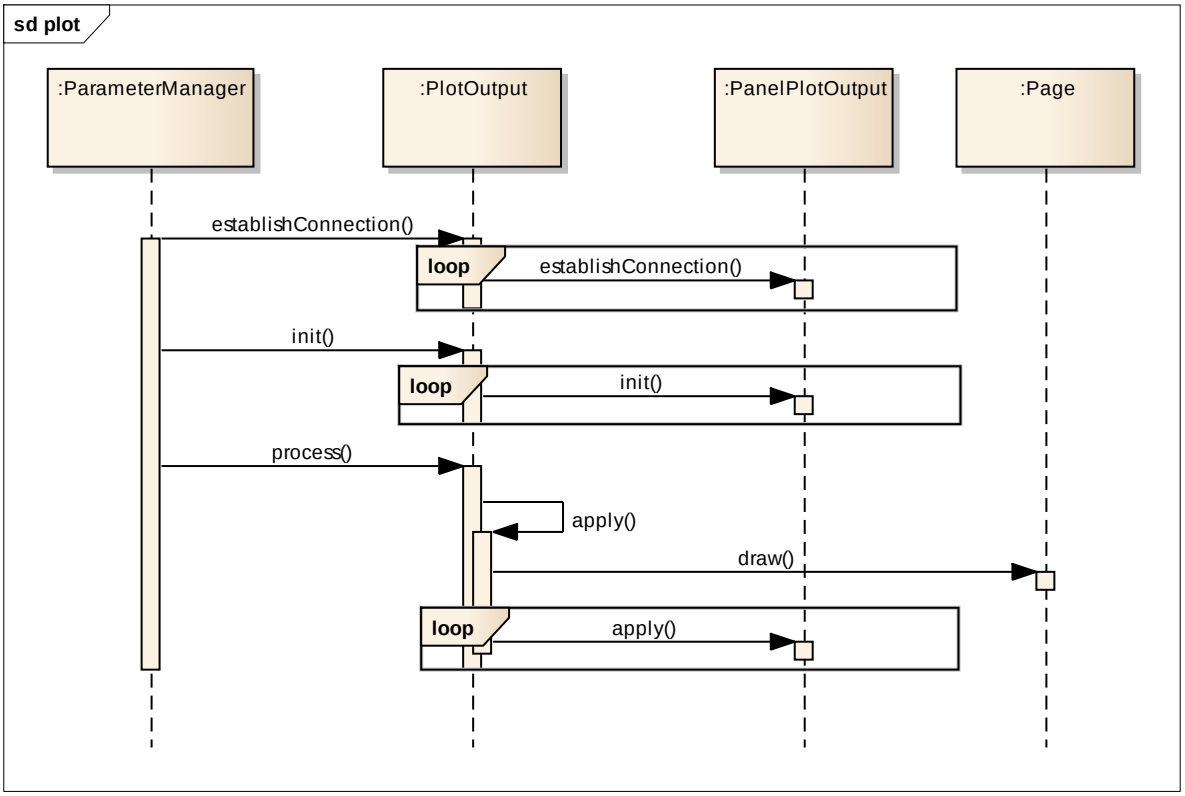


Figure 40 : Exécution d'une requête Plot

5.3.8.3.2 Types de visualisation

Les types de visualisation peuvent être regroupés en deux catégories : les tracés fonction du temps et les tracés fonction de paramètre. Des classes intermédiaires `TimePlot` et `XYPlot` sont ainsi ajoutées. Elles permettent de mutualiser des fonctionnalités et ainsi de faciliter l'ajout de nouveaux types de visualisation.

5.3.8.3.3 Classe PanelPlotNodeRegistry

Les différents types de visualisation sont répertoriés dans un registre `PanelPlotNodeRegistry` (même principe que pour le post-processing) à travers la classe responsable de la lecture du nœud XML relatif (implémentation de `AbstractPanelPlotNode`) et de l'instanciation de la classe de traitement (type de visualisation). Chaque classe de traitement doit enregistrer sa classe de lecture XML de manière statique.
Les mécanismes d'ajout de nouveaux types de plot sont présentés en annexe 2.

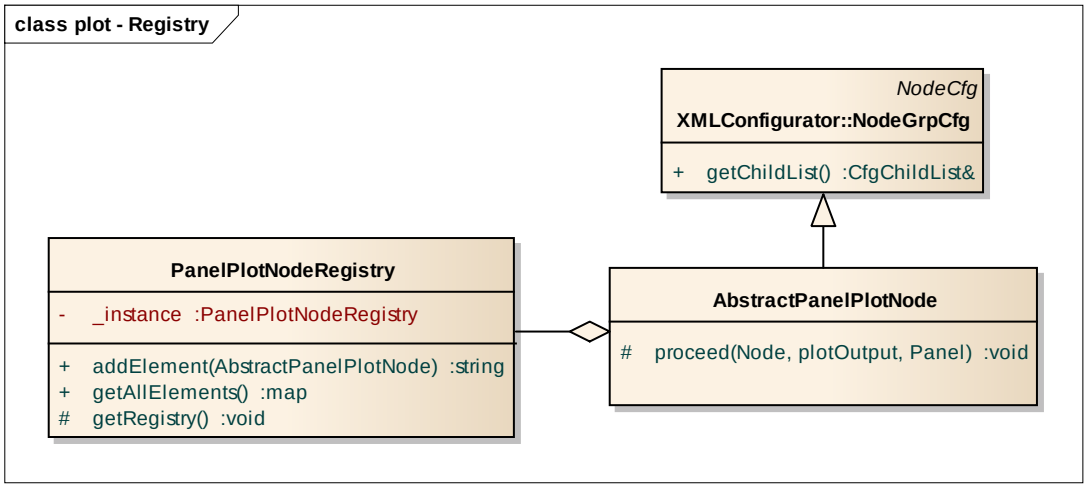


Figure 41 Classe PanelPlotNodeRegistry

Un mécanisme similaire est mis en place pour lire les différentes sections du fichier des valeurs par défaut (même registre, implémentation de `AbstractPlotConfigNode`) permettant ainsi de facilement ajouter des éléments de configuration propres à un type de plot.

5.3.8.4 Contexte de trace

Le contexte de tracé de tous les types de visualisation est organisé autour des notions de page, de panel et d'axe décrites ci-dessous.

Le schéma ci-dessous décrit l'organisation logique d'une page :

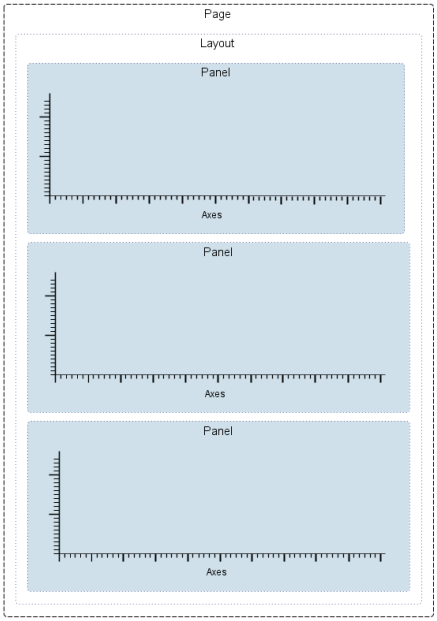


Figure 42 : Contexte du tracé (organisation logique)

5.3.8.4.1 Page

Une Page regroupe un ensemble de Panel et peut être associée à un Layout. Une page définit les caractéristiques suivantes :

- ✓ un format PNG, PS, SVG ou PDF
- ✓ une orientation paysage ou portrait (optionnel)
- ✓ une dimension A4 ou letter (optionnel)
- ✓ un mode couleur ou nuances de gris (optionnel)
- ✓ des marges verticales et horizontales (optionnel)
- ✓ une police d'écriture (optionnel)
- ✓ et un titre (optionnel)

Exemple de la requête XML (extrait) :

```
<outputs>
  <plot>
    <page format= "svg" dimension="US letter" orientation="landscape"
                                                                mode="color">
      <title position= "top" align= "left">Output format: svg, dimension: ISO
A4, orientation: landscape, font size: 12</title>
      <!-- panel definitions -->
    </page>
  </plot>
</outputs>
```

5.3.8.4.2 Panel

Un Panel représente une zone de tracé associée à un système d'axes. Un Panel peut être associé à plusieurs paramètres qui seront alors tracés sur le même graphe. Un panel définit les caractéristiques suivantes :

- ✓ une police d'écriture (optionnel)
- ✓ une couleur de fond (optionnel)
- ✓ un titre (optionnel)
- ✓ une position et une dimension (optionnel)

Exemple de la requête XML (extrait) :

```
<page format="png">
    <panel backgroundColor="[255,255,255]">
        <bounds x="0.2" y="0.2" width="0.6" height="0.6"/>
        <title>Panel title</title>
        <!-- plot definition -->
      </panel>
    </page>
  </plot>
</outputs>
```

5.3.8.4.3 Layout

Un Layout permet d'organiser les différents Panels dans la Page.
Non implémenté.

5.3.8.4.4 Axes

Les axes définissent une échelle physique et/ou temporelle des paramètres à tracer. Un axe peut être associé à plusieurs paramètres et définir les caractéristiques suivantes :

- ✓ une origine (optionnel, 0 si précisée), un axe parallèle est dessiné depuis le 0 de chacun des axes perpendiculaires
- ✓ une couleur (optionnel)
- ✓ une inversion (optionnel)
- ✓ une légende
- ✓ une position gauche, droite ou haut, bas (optionnel)
- ✓ une épaisseur de ligne (optionnel)
- ✓ des bornes min et max (optionnel), automatiquement positionnées à l'entier supérieur aux bornes min et max des paramètres lorsque l'utilisateur n'a pas positionné de bornes spécifiques
- ✓ un nombre de major ticks (optionnel), automatiquement ajusté à défaut
- ✓ un nombre de minor ticks (optionnel), calculé par plplot à défaut
- ✓ et une échelle linéaire ou logarithmique (optionnel)
- ✓ une marge (optionnel)

Il existe 4 types d'axe :

- ✓ Axe x : axe des abscisses portant des données de paramètres.
- ✓ Axe y : axe des ordonnées portant des données de paramètres.
- ✓ Axe t : axe des abscisses portant un temps,
- ✓ Axe epoch : axe des abscisses portant un temps relatif à un temps de référence '0'(non implémenté),

Les axes x et y possèdent une caractéristique propre qui permet de définir automatiquement les bornes en fonctions des valeurs des paramètres (optionnel).

Il est possible pour l'axe t à de définir le format de la date :

- ✓ jour/mois/année(dd/mm/yy), (défault)
- ✓ numjour/année (ddd/yy).

Les axes sont séparés en deux grandes séries :

- ✓ Les axes temporels (`TimeAxis`) qui sont uniquement utilisables en abscisse,
- ✓ Les axes numériques (`DigitalAxis`) qui sont utilisables en ordonnée et en abscisse.

Exemple de la requête XML (extrait) :

```
<panel>
  <timePlot>
    <params/>
    <axes>
      <xAxis>
        <timeAxis position="bottom" thickness="1" format="ddd/yy">
          <tick majorSpace="14400" minorSpace="3600" /> <!-- in seconds -->
        </timeAxis>
      </xAxis>
      <yAxis>
        <digitalAxis id="1" origin="0" position="left" reverse="0"
              thickness="1" autoScale="false" scale="linear" color="[255,0,0]">
          <range min="0" max="100" extend="false" />
          <tick majorNumber="5" minorNumber="5" lengthFactor="1"/>
        </digitalAxis>
      </yAxis>
    </axes>
  </timePlot>
</panel>
```

5.3.8.4.5 Classes du contexte

Le diagramme ci-dessous présente les classes de ce contexte ainsi que leur organisation.

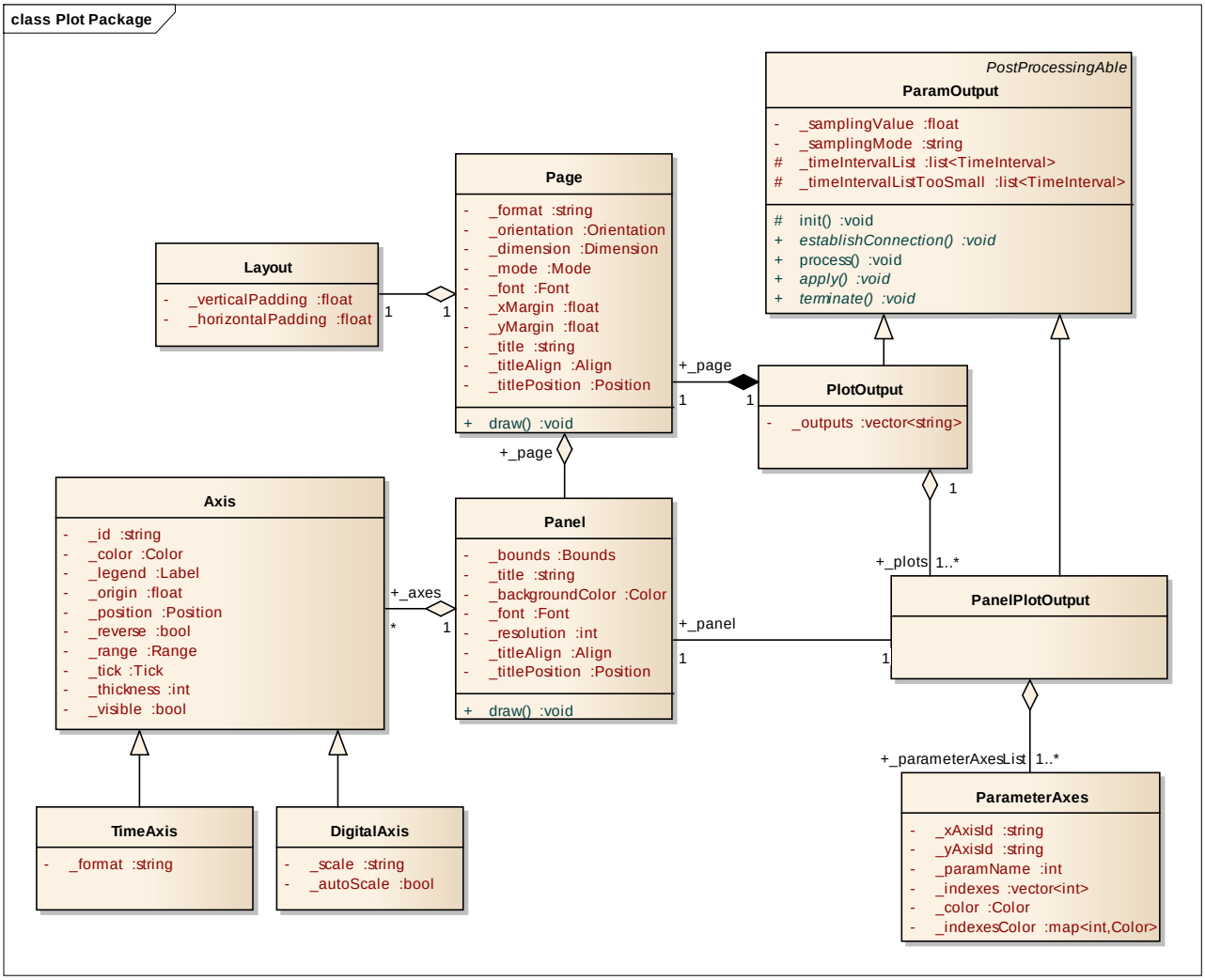


Figure 43 : Classes principales du contexte d'une requête de Plot

5.3.8.4.6 Fichier plotConfig.xml

Il existe un fichier de configuration au format XML `plotConfig.xml` dont le chemin est spécifié dans le fichier `app.properties` positionnant :

- ✓ les valeurs par défaut des propriétés du contexte ainsi que les valeurs imposées par le système. Lors de la création d'une instance de `Page` de `Panel` ou d'un `Axis`, les propriétés relatives sont initialisées avec les valeurs imposées/par défaut et sont ensuite surchargées lorsqu'elles sont redéfinies par l'utilisateur dans la requête.
- ✓ la définition des palettes de couleur utilisables dans les requêtes de plot via l'attribut `colorMapIndex`.

Ce fichier est modifiable par l'administrateur du système.

Les diagrammes ci-dessous présentent les classes responsables de la gestion des valeurs par défaut et des palettes de couleur.

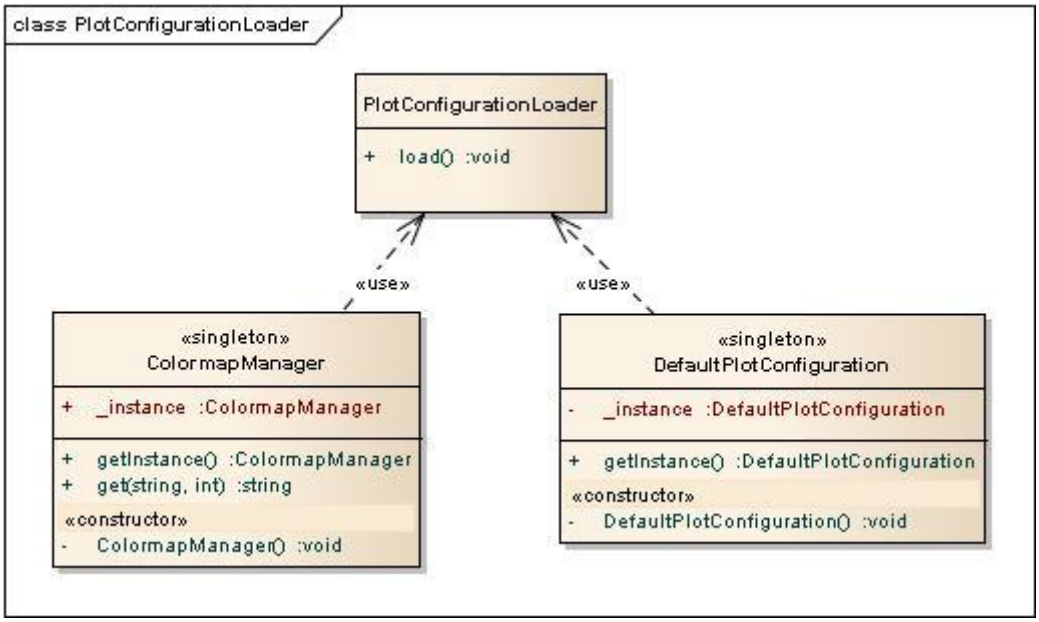


Figure 44 : Classe PlotConfigurationLoader

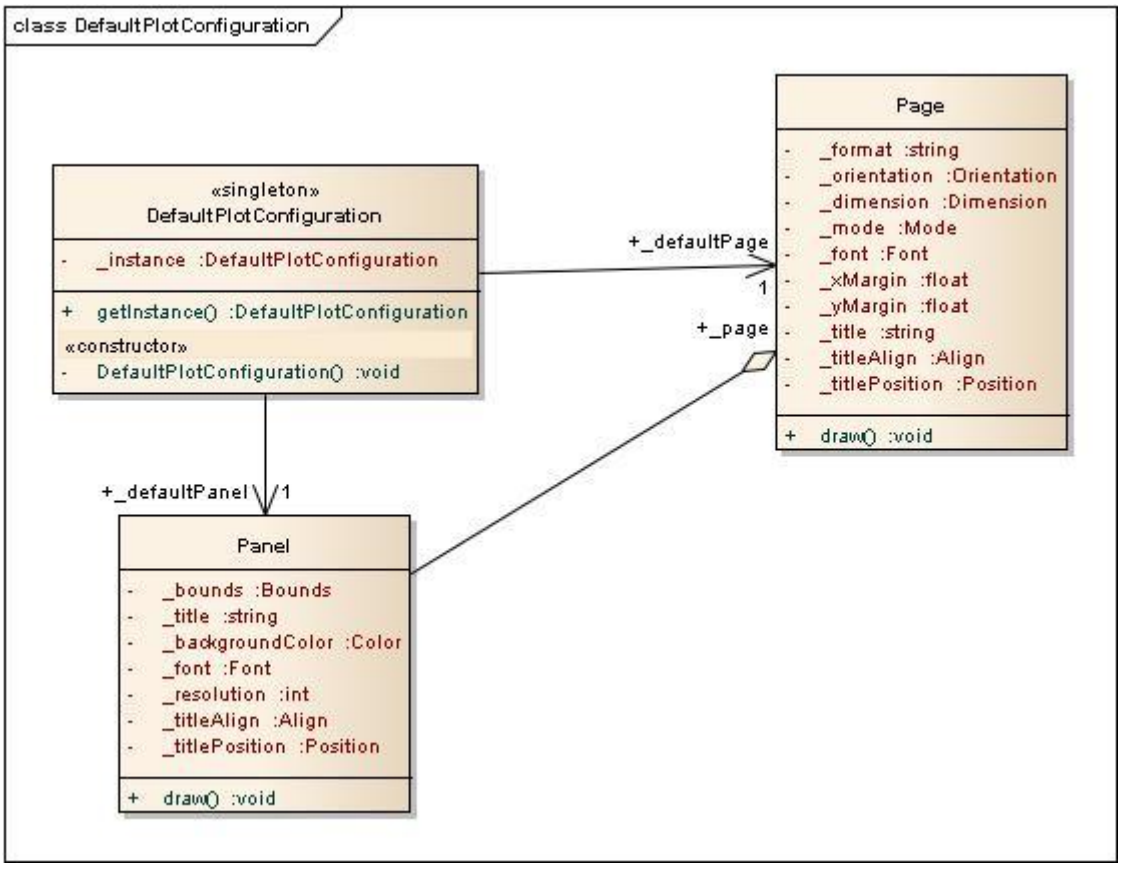


Figure 45 : Classe DefaultPlotConfiguration

5.3.8.5 Paramètres à tracer, notion de serie

La gestion des paramètres est effectuée au niveau de la classe `PanelPlotOutput` via une liste de `ParameterAxe`.

La notion de `serie` est introduite pour définir "l'ensemble des mesures rattachées à une composante d'un paramètre et qui sera tracé sur le graphe".

- ✓ une `serie` porte les caractéristiques de dessin (type de courbe, type de symbole, couleur).
- ✓ une `serie` est rattachée à un paramètre ou à une composante de paramètre (définition d'un index pour les paramètres de type vecteur)
- ✓ une `serie` est rattachée à un axe X (axe du temps dans le cas d'une série temporelle) et un axe Y

Exemple de la requête XML (extrait) :

```
<panel>
  <timePlot>
    <params>
      <param id="imf">
        <serie index="0" yAxis="1">
          <line style="plain" width="2" color="[10,120,13]"/>
          <symbol type="*" size="3" color="[10,120,13]"/>
        </serie>
      </param>
    </params>
    <axes/>
  </timePlot>
</panel>
```

Il est possible, pour un même paramètre de définir un ensemble de caractéristiques communes aux séries relatives (une série par composante) par le biais d'une section `default`.

Dans l'exemple qui suit, les composantes 1 et 2 sont tracées avec les caractéristiques définies par défaut pour l'ensemble des séries alors que la composante 0 définit un axe Y, une couleur et des types de ligne et symbole propres

Exemple de la requête XML (extrait) :

```
<panel>
  <timePlot>
    <params>
      <param id="imf">
        <default yAxis="..." xAxis="..."> <!-- default yAxis and/or xAxis -->
          <line style="plain" ..../> <!-- default line properties -->
          <symbol type="no" ..../> <!-- default symbol properties -->
        </default>
        <serie index="0" yAxis="1">
          <line color="[0,33,00]" style="dot"/>
          <symbol type="plus"/>
        </serie>
        <serie index="1"/>
        <serie index="2"/>
      </param>
    </params>
    <axes/>
  </timePlot>
</panel>
```

Pour chaque série, il est possible de définir un nombre maximum de points par plot via un attribut resolution.

Exemple :

```
        <serie index="1" resolution="1000"/>
```

Le paramètre ou la composante de paramètres est alors ré-échantillonné, si nécessaire, pour obtenir, sur la période de la requête, au maximum, le nombre de points précisé par l'attribut `resolution`. Lorsque cet attribut est positionné à -1, le sampling naturel du paramètre est conservé.

Les caractéristiques graphiques d'une série sont portées par la classe `DrawingProperties`. Il existe 2 types de série : les séries en y (classe `SeriesProperties`) et les séries en x (classe `XSeriesProperties`).

5.3.8.6 Types de plot

5.3.8.6.1 Time plot

La réalisation d'un tracé f(t) est portée par la classe `TimePlot`. La requête relative est lue par la classe `TimePlotNode` et la section de valeur par défaut dans le fichier de configuration du module plot par la classe `TimePlotConfigNode`.

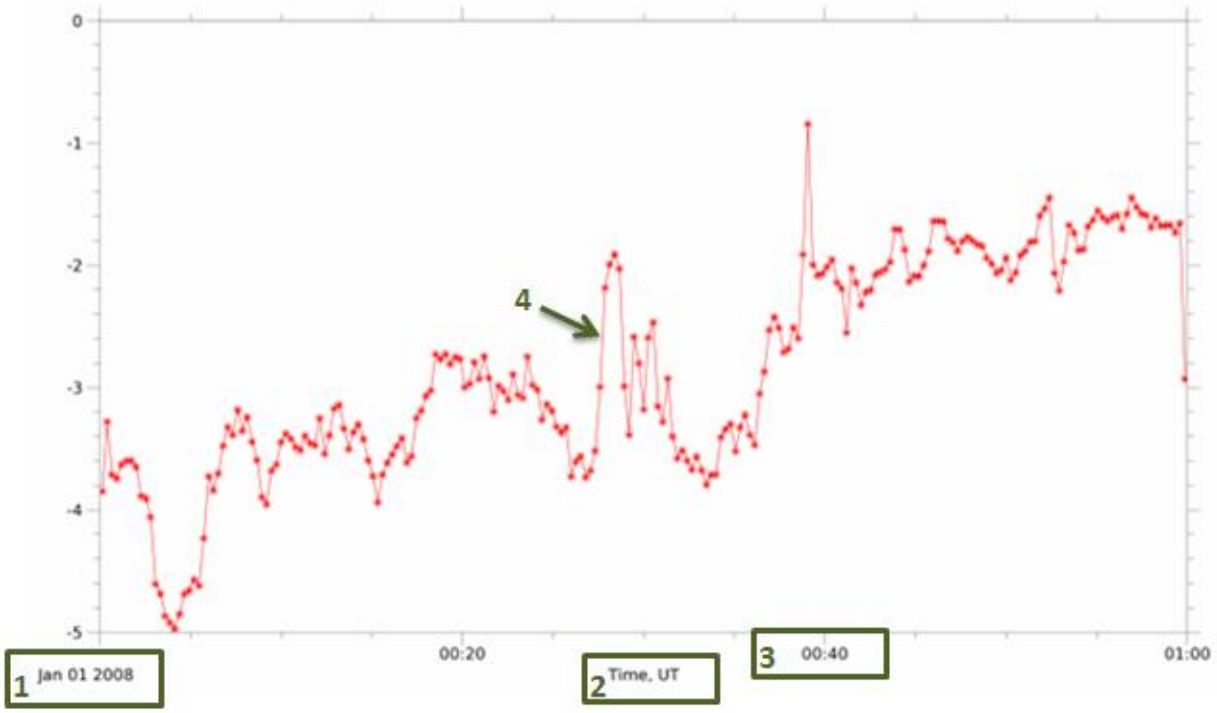


Figure 46 : Tracé time plot

(1) La start date de la requête est affichée en bas à gauche (en bas à droite dans le cas d'un axe du temps reverse).
(2) La légende de l'axe du temps est positionnée par défaut dans plotConfig.xml, il s'agit d'un `Label`.
(3) Le format des libellés des ticks majeurs est automatiquement adapté à l'intervalle de temps tracé et au nombre de ticks majeurs. De plus, le libellé du premier tick majeur n'est pas affiché lorsqu'il correspond à l'origine de l'axe t. La gestion du formatage des libellés est assurée par la classe

`DefaultTimeAxisGenerator` qui exploite la fonction `plslabelfunc` de plPlot pour déclarer un générateur de libellés customisé.

(4) Les types de courbe et de symbole ainsi que leurs couleurs sont configurable dans la requête.

5.3.8.6.2 Scatter plot

La réalisation d'un tracé f(x) est portée par la classe `XYPlot`. La requête relative est lue par la classe `XYPlotNode` et la section de valeur par défaut dans le fichier de configuration du module plot par la classe `XYPlotConfigNode`.

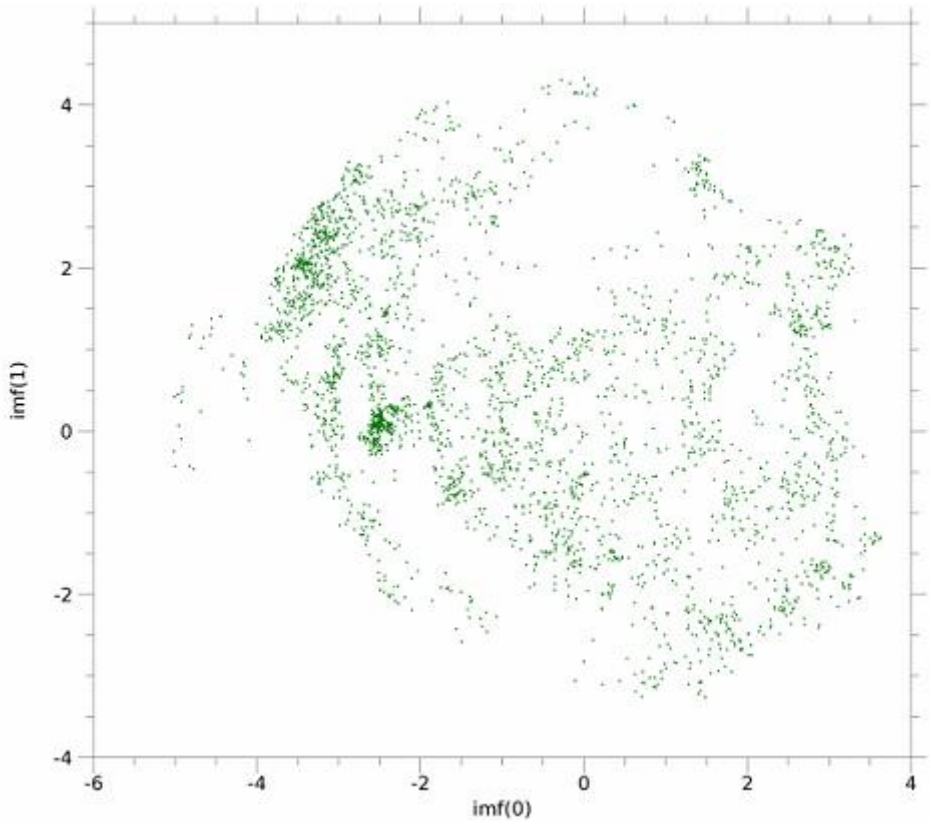


Figure 47 : Tracé scatter

Pour chaque association xy, le paramètre en abscisse et le paramètre en ordonnée sont ré-échantillonnés sur la période de la requête avec comme date de référence la date de début de l'intervalle. On obtient ainsi deux tableaux de valeurs qui sont passés à plplot pour tracer les points et, éventuellement, les lignes entre les points.

Il existe par ailleurs un attribut `isotropic` qui permet de forcer le ratio entre l'axe x et l'axe y pour obtenir un tracé tel que celui présenté ci-dessus.

5.3.8.6.3 Tickmark plot

La réalisation d'un tracé tickmark est portée par la classe `TickPlot` (qui hérite de `TimePlot`). La requête relative est lue par la classe `TickPlotNode`. Ce type de tracé a la particularité de pouvoir être associé à un tracé f(t) et partager le même axe du temps.

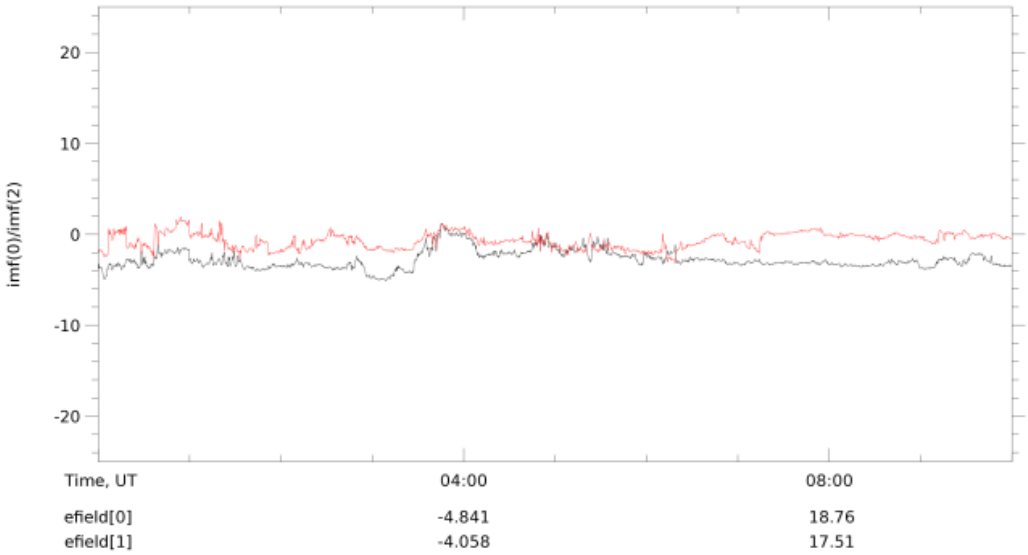


Figure 48 : Tracé tickmark associé à une série temporelle

Time, UT	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00
speed	93.46	175	240.8	283.9	300.1	287.9	248.4	185.6	105.7	23.98
efield[0]	NAN	NAN	0.887	-4.841	1.593	6.172	9.682	18.76	NAN	NAN
efield[1]	NAN	NAN	-5.268	-4.058	-2.819	-0.4025	-18.75	17.51	NAN	NAN
efield[2]	NAN	NAN	-5.169	-3.942	16.91	-2.398	-5.946	-10.08	NAN	NAN

Figure 49 : Tracé tickmark indépendant

Les valeurs des paramètres du tickplot sont affichées sous les ticks majeurs de l'axe du temps. Lorsqu'il n'y a pas de valeurs pour un temps t donné, c'est la valeur la plus proche dans le temps qui est affichée.

Au niveau de la requête, ceci se traduit de la façon suivante :

Premier cas : le tickplot est associé à une série temporelle, il est décrit à l'intérieur du même panel. C'est l'axe du temps du timeplot qui est utilisé.

```
<panel>
  <timeplot>
    <!-- description des paramètres du time plot et de l'axe du temps -->
  </timePlot>
  <tickPlot>
    <params>
      <param id="efield">
        <serie index="0"/>
        <serie index="1"/>
      </param>
    </params>
  </tickPlot>
</panel>
```

Second cas : le tickplot est indépendant, il décrit son propre axe du temps (toujours invisible) et notamment l'espacement des ticks.

```
<panel>
 <tickPlot>
   <params>
     <param id="speed">
       <serie/>
```

```
    </param>
    <param id="efield">
      <serie index="0"/>
      <serie index="1"/>
      <serie index="2"/>
    </param>
  </params>
  <axes>
    <xAxis>
      <timeAxis position="bottom"  thickness="1">
        <tick majorSpace="3600" />
      </timeAxis>
    </xAxis>
  </axes>
 </tickPlot>
</panel>
```

Qu'ils soient associés ou non à une série temporelle, les tickmarks sont gérés par un décorateur spécifique `TickMarkDecorator` associé à l'axe de temps qui permet de gérer l'affichage des valeurs des paramètres sous chaque tick majeur de l'axe de temps. Lorsqu'il s'agit d'une association avec TimePlot le décorateur par défaut (`DefaultTimeAxisDecorator`) est simplement remplacé par le décorateur spécifique.

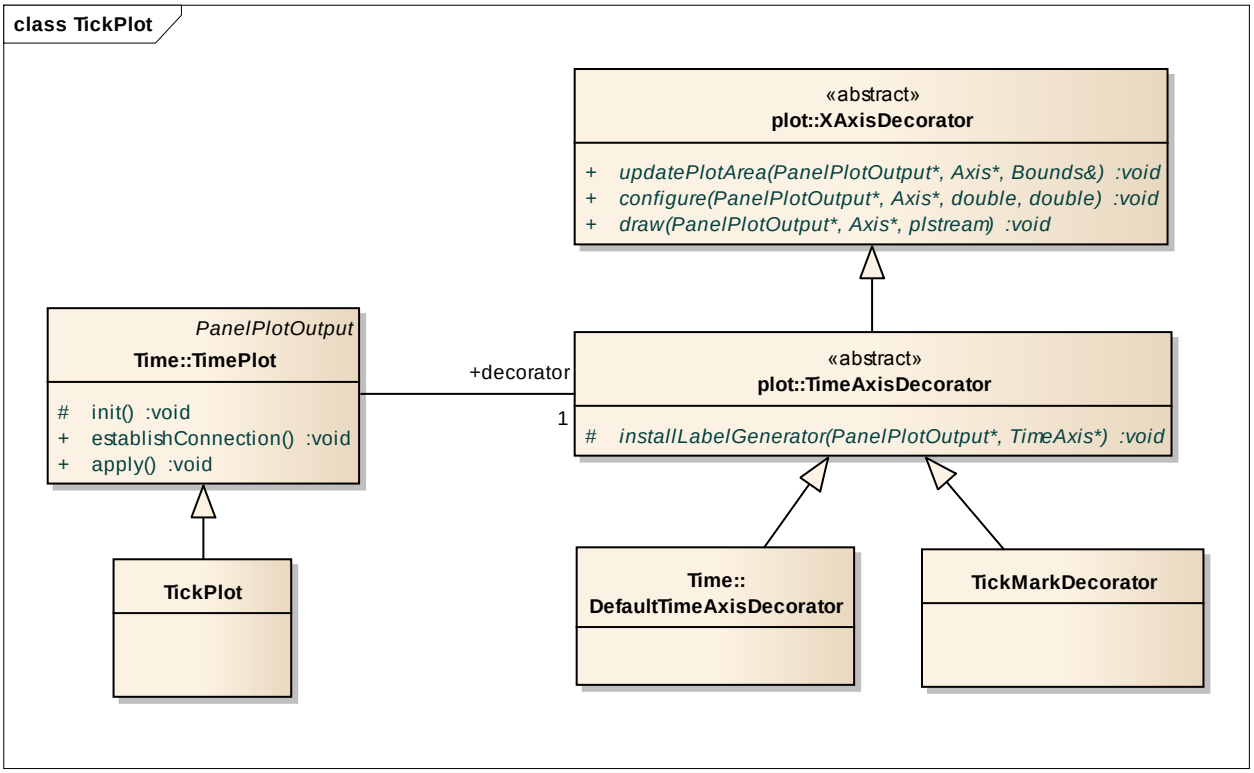


Figure 50 : TimeAxisDecorator

L'élément `tickPlot` prend en compte un attribut `format` qui peut être utilisé pour spécifier le formatage des valeurs affichées. Ce format utilise les spécifications de `printf` (cf. http://www.cplusplus.com/reference/cstdio/printf/ pour une description détaillée), en voici le prototype :

`%[flags][width][.precision][length]specifier.`

A défaut de précision par l'utilisateur, le format suivant est utilisé : % -4.4G.

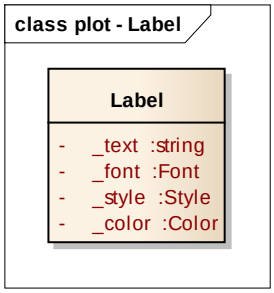
- ✓ flags = <space>- : Les valeurs affichées sont justifiées à gauche et un espace est affiché en lieu et place du signe pour les valeurs positives (permet un alignement des valeurs)
- ✓ width = 4 : Affiche au minimum 4 chiffres, lorsque le nombre ne contient pas 4 chiffres, les chiffres manquants sont remplacés par des blancs
- ✓ .precision = .4 : Nombre de chiffres à afficher après la virgule
- ✓ specifier = G : Affiche la valeur en notation scientifique lorsque celle-ci est plus courte que la notation flottante.

5.3.8.7 Objets Additionnels

Des objets additionnels peuvent être associés à un panel.

5.3.8.7.1 Label

Il s'agit d'un texte affiché dans la zone de plot et définissant le contenu du texte, sa couleur , sa police d'écriture, son style (normal, gras, italique) et son orientation (horizontal, vértical).



5.3.8.8 Dépendances

Le composant Plot contient plusieurs implémentations de `ParamOutput`.

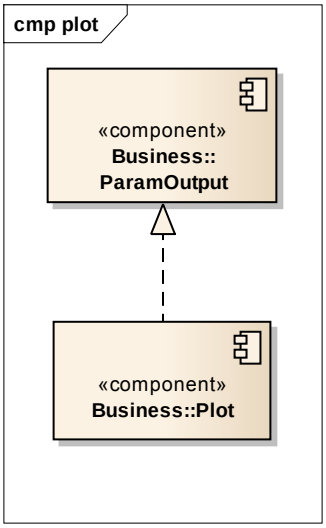


Figure 51 : Dépendances du composant Plot

ANNEXE 1 – Ajout d'un post-traitement

1/ Ajout du nœud `<new-post-process>` dans le fichier descriptif de requête `postProcessing.xsd`

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <xs:include schemaLocation="download.xsd" />

    <xs:element name="postProcess" substitutionGroup="PostProcessingElement" >
            <xs:complexType>
            <xs:sequence>
                <xs:element name="zip" minOccurs="0" maxOccurs="1"/>
                <xs:element name="tar" minOccurs="0" maxOccurs="1"/>
                <xs:element name="gzip" minOccurs="0" maxOccurs="1"/>
                <xs:element name="newPostProcess" minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
        </xs:complexType>
      </xs:element>

</xs:schema>
```

2/ Implémentation `<new-post-process>Node` de la classe `XMLConfigurator::NodeCfg` pour lire le nœud `<new-post-process>`

```
class NewPostProcessNode: public AMDA::XMLConfigurator::NodeCfg {
public:
      NewPostProcessNode () : NodeCfg() { }
      virtual ~NewPostProcessNode () {}

      void proceed(xmlNodePtr pNode,
                  const AMDA::Parameters::CfgContext& pContext){

          PostProcessingAble* output = pContext.get<PostProcessingAble*>();

          // Set value in output param
          output->addPostProcessing(new NewPostProcessPostProcessing);
      }

};
```

3/ Implémentation de l'interface `PostProcessing`, méthodes `apply()` et `registerChildList()`

```
class NewPostProcessPostProcessing: public PostProcessing {
public:
      NewPostProcessPostProcessing ();
      virtual ~NewPostProcessPostProcessing ();

      /**
       * Applies post-processing.
       */
      void apply(PostProcessingAble* pOutput){
           // do something
      }
```

```
      /**
       * Registers related node and process.
       */
      void registerChildList(std::map<std::string,
                boost::shared_ptr<AMDA::XMLConfigurator::NodeCfg>>& childList){

          childList["newPostProcess"]=
                boost::shared_ptr<AMDA::XMLConfigurator::NodeCfg>(
                                                    new NewPostProcessNode());
      }

private:
      /**
       * Attribute to force self-registering to post-processing factory
       */
      static std::string _key;
};
```

4/ Enregistrement statique de la classe d'implémentation de `PostProcessing` auprès de `PostProcessingRegistry`.

```
/**
 * Self-registering to post-processing factory
 */
std::string NewPostProcessPostProcessing:: key =
            PostProcessingRegistry::getInstance().addElement("newPostProcess",
                        new GzipPostProcessing());
```

5/ [Optionnel] Complétion de la classe `PostProcessingAble` si le post-traitement nécessite des informations particulières sur l'output

ANNEXE 2 – Ajout d'un type de plot

1/ Implémentation <new-plot>, cette classe peut dériver de `TimePlot`, `XYPlot` ou directement de `PanelPlotOutput`.

```
class NewPlot: public TimePlot {

public:
      NewPlot (AMDA::Parameters::ParameterManager& manager,
                  boost::shared_ptr<Panel> panel);
      virtual ~NewPlot ();

      // […] implement apply method to draw plot
      // […] optionnaly implement visitor methods

};
```

2/ Implémentation `<new-plot>Node` de la classe `Plot::AbstractPanelPlotNode` pour lire le nœud `<new-plot>`

```
class NewPlotNode: public AbstractPanelPlotNode {

public:
      NewPlotNode () : AbstractPanelPlotNode () {
          // register child nodes
          getChildList()["params"] = AMDA::XMLConfigurator::NodeCfgSPtr(
                  new ParamsNode<TimePlot>());
          getChildList()["axes"] = AMDA::XMLConfigurator::NodeCfgSPtr(new
                                          AxesNode());
      }
      virtual ~ NewPlotNode () {}

      boost::shared_ptr<PanelPlotOutput> proceed(xmlNodePtr
                        pNode_, PlotOutput* plotManager_, Panel* panel_){
          // create plot type treatment class
          boost::shared_ptr<NewPlot> plot(
                  new NewPlot(plotManager_->getParameterManager(),
                              boost::shared_ptr<Panel>(panel_)));
          // fill context for child nodes
          AMDA::Parameters::CfgContext context;
          context.push<NewPlot*>(plot.get());
          context.push<Panel*>(panel_);

          // process child nodes
          AMDA::XMLConfigurator::NodeGrpCfg::proceed(pNode_, context);

          return plot;
      }

private :
      /**
       * constant to uniquely identify this node in the registry
       */
      static const std::string NODENAME;

      /**
       * artificial special attribute to force registering node.
```

```
        */
       static std::string _key;

};

/** Static registration in .cc **/
std::string NewPlotNode::_key =
PanelPlotNodeRegistry::getInstance().addElement( NODENAME,
            boost::shared_ptr<NewPlotNode>(new NewPlotNode()));
```

Versions successives

Version	Date	Émetteur	Vérificateur	Approbateur	Motif
1.6	23/01/2014	N.Boursier	S.Frayssines	R.Patrier	Corrections tickplot et f(x) suite aux remarques fin de sprint 5
1.5	10/01/2014	N.Boursier	S.Frayssines	R.Patrier	Traitement des User Stories du sprint 5.
1.4	28/11/2013	N.Boursier	S.Frayssines	R.Patrier	Traitement des User Stories du sprint 4.
1.3	08/11/2013	N.Boursier	S.Frayssines	R.Patrier	Traitement des User Stories du sprint 3.
1.2	08/10/2013	N.Boursier	S.Frayssines	R.Patrier	Traitement des User Stories du sprint 2.
1.1	16/09/2013	N.Boursier	S.Frayssines	R.Patrier	Prise en compte des FEBS : 5 et 6 Traitement des User Stories du sprint 1.
1.0	26/07/2013	N. Boursier	S.Frayssines	R.Patrier	Création du document

Diffusion

Ce document est mis à disposition sous forme informatique sur serveur.
Il n'est donc pas formellement diffusé sous forme papier.