Lexique :

expression	définition
`basedir`	Le répertoire dans lequel sont installés les programmes cafeterra. Il est vivement conseille d'utiliser le répertoire suivant : `/home/app/cafeterra`
Contexte	Un contexte est la définition de l'environnement de travail dans cafeterra. Il y a au moins deux environnement de travail, qui correspondent aux différents type d'environnement de travail : Développement (ou Design) et Production. L' environnement de développement correspond à un contexte implicite, il n'est pas nécessaire de le définir dans la base de donnée cafeterra. Chaque environnement de travail correspond à un répertoire et une base de donnée Postgres. par exemple le contexte de développement possède le répertoire de travail `/home/app/cafeterra/cgi` et une base de données, que l'on conseille d'appeler `cafeterra`. Le contexte `default` installé par défaut correspond répertoire `/home/app/cafeterra/cgi` et une base de données, appelée `cafeterraq`. Pour créer un contexte il faut : - d'abord utiliser la commande `perl /home/app/cafeterra/install/Pg/install.pm`, - ensuite, il faut enregistrer la définition de ce contexte dans l'environnement de développement et dans sa base de données.
Utilisateur Cafeterra	Il s'agit des utilisateur autorisé à développer des flux (environnement de développement) ou les administrer (environnement de production).
Utilisateur	Identifiant utilisé par Cafeterra pour se connecter sur les systèmes externes
Serveur	Définition d'un serveur (Adresse IP ou non DNS etc ..)
Protocole	Pour être simple, disons qu'il s'agit des mécanisme qui permettent à cafeterra de se connecter à la base de donnée ou à l'application concerné.
Pilote (ou driver)	Le pilote décrit comment traduire les données échangées dans un langage compréhensible par l'application ou la base de données à laquelle Cafeterra est connecté.
Conteneur	Il s'agit de la définition précise de l'emplacement ou les données vont lues ou écrite. Par exemple une le nom Table Oracle, le nom d'un Fichier CSV, etc ... La définition inclut les éléments suivant :

	- Nom interne (local) - Nom externe nom de la Table ou la vue oracle, Nom du fichier CSV, dans le cas d'un fichier EXCEL, ce sera le nom de la feuille de calcul - Connecteur (ou base de donnée dans laquelle se trouve ce conteneur) - Direction de flux privilégié (Entrant, Sortant, les deux) - Doit on utiliser les espaces Rollback pour ce conteneur, Voir Espace Rollback - plus un certain nombre d'information en fonction du protocol et du pilote utilisé
Flux	Définition d'un flux de données entre deux ou plusieurs applications La définition inclut les éléments suivant : - Nom interne (local) - Type du flux (simple, ou Web Service) - Sous flux indirect = dans ce cas Cafeterra joue le rôle d'an tampon entre l'application source et l'application destinatrice, le flux, de ce fait, sera historisé. Autrement, les données seront transmises de la sources à la destination en temps réel. - Dépendance entre les flux entrants (quand il y en a plusieurs) - plus un certain nombre d'information en fonction du protocole et du pilote utilisé Outre ceci, on peut définir pour un flux un certain nombre de traitement à effectuer, au début ou à la fin de son execution, ou en cas d'erreur. On peut aussi, adjoindre au flux d'autres objets (Scripts, Connecteurs, Conteneurs, accessibles aux scripts cafeterra d'une manière simple) Remarques : - Le fonctionnement normal de Cafeterra est de délivrer les message entrant dès leur arrivée. Dans le cas où, les applications destinatrices ne sont pas disponibles, et si le flux est `flow`, Alors cafeterra enregistre les messages non délivrés, et réessayera de les délivrer plus tard. - Cafeterra peut garder dans son historique 7 jours d'activité au maximum. Si vous avez décidé d'archiver vos message, ce peut être indéfini. - En général, on utilise des flux entrant dépendant quand on veux faire des jointures entre plusieurs Connecteurs différents, mêmes ceux qui ne supportent pas les jointures. Par exemple, on peut faire une jointure entre une table Oracle, un fichier XML, et un fichier EXCEL !!
Demi-Flux	Description de l'application ou de la base de données emmetrice

Entrant	d'information. Il s'agit d'une définition dynamique du flux. Pour chaque demi flux entrant on associe un conteneur. La méthode de récupération des donnes, défini s'il s'agit d'un flux de message au fil de l'eau, ou si à chaque une les messages remplacent les messages précédent. Ceci est utile dans le cas des flux historisés. Un sous flux est associé à un ensemble de traitement dont un seul est obligatoire : C'est la requête SQL qui permettra de récupérer les données sources(`getmsg`). On peut éventuellement définir une action qui permettra d'envoyer un accusé réception du message. D'autre scripts seront execute avant après lectire de chaque message etc ... Le mapping permet de transformer des champs récupéré dans le message. ou initialiser des champs avec des valeurs fixes.
Demi-Flux Sortant	Description de l'application ou de la base de données destinatrice d'information. Il s'agit d'une définition dynamique du flux. Pour chaque demi flux sortant on associe un conteneur. Le message peux déclencher soit l'ajout d'information dans l'application destinatrice, soit la modification de d'information ou la suppression préexistante (`getmsg`). On peux définir deux action `query`. Si la première échoue, la seconde requete SQL sera executé. D'autre scripts seront execute avant après lectire de chaque message etc ... Le mapping permet dans l'ordre : 1 - Affecter un champs en entrée à un champs en sortie, 2 - Affecter une constante ou le contenue d'une variable au champs en sortie, 3 - Affecter le résultat d'un script au champs en sortie. Le scripts de transformation global permet d'effectuer d'autres transformation
Requete SQL	SQL a été adopté pour accéder à toute forme de données y compris le CSV, XML, HTML etc ... Les différents type de requete sont : 1 - Select 2 - Insert 3 - Update 4 - Delete 5 - truncate 6 - Drop

	7 - Create Il est nécessaire bien renseigner le champs utilisé pour (`usedfor`) pour que cafeterra puisse décider quel requete utiliser quand. On peut écrire une requête manuellement. Mais il est conseillé d'utiliser l'assistant cafeterra pour la création d'une requête. Une syntaxe particulière doit être utilisé. ainsi - tout identifiant commença par `@c_, @i_ ou @o_` est considéré comme un alias du champs en question. Ceci est valable même pour les bases de données qui ne supportent pas l'aliasing dans les requêtes. - tout identifiant commençant par `:c_, :i_, :o_ , :s_ , :f_, :g_ ou :t_` sera remplacé (avant exécution de la requête) par le contenue de la variable Cafeterra de même nom. Il correspondent aux fameux `bind variables` - tout identifiant commençant par `$c_, $i_, $o_ , $s_ , $f_, $g_ ou $t_` sera remplacé par le contenue de la variable Cafeterra correspondante avant soumission de la requête au pilote adéquat (avant le parsing). Ceci peut être utile quand on ne connaît pas le nom de la table au moment du développement de la requête.
Script Perl	Comme vous l'avez vu, vous pouver définir des traitement tout au long de l'execution d'un Flux. Ces script sont écrit en Perl. Ces script reçoivent un paramètre, par convention on l'appelle `$self`. Il permet d'accéder au fonction cafeterra et au variables des Flux. Peu de contrainte sont imposé aux scripts perl : - Les scripts de transformation de champ doivent renvoyer une valeur qui ser affecté au champ en question. - Le script de gestion d'erreur reçoit le texte de l'erreur en paramètre supplémentaire et doit renvoyer une des valeur chaîne suivante : - "HALTE" - Arrêter le flux global, il ne sera plus exécuté, jusqu'à l'intervention de l'administrateur - "STOP" - Arrêter le demi-flux qui a provoqué l'erreur, il ne sera plus exécuté, jusqu'à l'intervention de l'administrateur. Le flux global continue lui à être executé. Ce retour n'est pas autorisé, dans le gestionnaire d'erreur associé au flux global - "SKIP" - Ignorer l'erreur, si un message est en cours de traitement, il sera abandonné, même dans le cas d'un flux historisé. - "RETRYLATER" - Ignore l'erreur, Si un message en en cours de traitement,il sera marqué comme non traité, et le sera dès que possible, à condition que le flux global soit historisé (Flux indirect) Consulter dans le ficher functionlist.txt, la liste des fonction

	disponible au traver de l'objet . L'object `$self` vous permet d'accéder à vos variable et les champs de vos conteneurs avec la syntaxe suivante : - `$self->X_nomvar` pour lire le contenue de la variable - `$self->X_nomvar(VALEUR)` pour affecter une valeur à votre variable, utiliser `undef` pour effacer la variable X est un caractère parmi les suivants : - c pour champs de conteneur - i pour champs de conteneur dans le flux entrant - o pour champs de conteneur dans le flux sortant - s pour une variable accessible uniquement au demi flux en cours d'exécution - f pour une variable accessible uniquement au flux global et à l'ensemble des demi-flux du flux en cours d'exécution - g pour une variable global accessible à tous les flux - t -- voir f Remarque : les variable s, f et g sans sauvegardé, et sont restauré d'une execution à l'autre. les autres on une duré de vie limité à la duré de vie du programme (pour le t) et la durée de vie du message (por les c, i et o).
Variable Cafeterra	Voir ci-dessus Il n'est pas nécessaire de déclarer des variables dans Cafeterra, le fait d'y affecter une valeur, provoque sa déclaration et son initialisation. Néanmoins, ils est possible pré déclarer des variable dans la définition d'un flux GLOBAL.
Variable d'environneme nt	A l'execution d'un flux, il est possible d'initialiser des variable d'environnement système.
Evenement	Il s'agit d'un mécanisme qui permet de synchroniser l'execution des flux. Ainsi on peut subordonner l'exdecution d'un flux A à l'execution d'un Flux B et d'un Flux B.
Aliasing des connecteurs	Si vous travaillez correctement, vous devez avoir au moins 3 environnement : - Un environnement de développement - Un environnement de test - Un environnement de production Vos flux, avant d'être mis en production, va être testé dans l'environnement TEST. Ce qui veux dire que votre flux doit s'exécuter correctement dans deux environnements différents ce qui signifie : - Des serveurs différents, - Des login différents,

	- ou des bases données différentes Pour vous éviter de créer deux flux identique qui vont s'exécuter dans deux environnement différents, Cafeterra vous donne la possibilité de créer des Alias pour vos connecteur. Ainsi, quand vous déployer un connecteur dans un environnement particulier, Cafeterra cherchera d'abord, sir pour ce contexte, vous aver défini alias pour ce connecteur, si c'est le cas, c'est ce dernier qui va être déployé.
Traçabilité	L'historisation des messages dans Cafeterra joue deux Rôles : 1 - D'abord elle sert de tampon entre les messages entrants et les messages sortant, 2 - Ensuite elle offre pour les entreprises qui le souhaitent, une traçabilité de leurs flux.
Espace Rollback	La plupart des flux provoquent une le remplacement d'informations existantes. Dans certains cas ce peut être préjudiciable. Parce que ce type de problème est incontournable, Cafeterra offre la possibilité un mécanisme qui permettra d'annuler les effet d'un flux. Pour ce faire, Cafeterra récupérera le contenu d'une table dans Cafeterra, avant de délivrer les messages entrant. Ce mécanisme, renforce la traçabilité offerte par le mécanisme d'historisation des flux, et vous permet de vous prémunir contre les erreurs inévitables dans un environnement informatique complexe et presque imprévisible.
Archivage	Pour garder une file des messages la plus légère possible pour des raison de performance, Cafeterra supprimes toutes les information qui ont plus de 7 jours. Si vous le souhaitez, il possible d'indiquer à Cafeterra dans quelle base de données, ces informations seront archivées avant d'être supprimées.
Planification	Cafeterra possède son propre planificateur de tache. Il est inspiré du Cron Unix. La granularité des exécution est de l'ordre de la minute. Un flux pour posséder plusieurs planifications.