

Personnalisation d'un serveur EOLE à l'aide de Creole

EOLE 2.5.2



EOLE 2.5.2

Version : révision : Avril 2018

Date : création : Mai 2015

Editeur : Pôle national de compétences Logiciels Libres

Auteur(s) : Équipe EOLE

Copyright : Documentation sous licence Creative Commons by-sa - EOLE
(<http://eole.orion.education.fr>)

Licence : Cette documentation, rédigée par le Pôle national de compétences Logiciels Libres, est mise à disposition selon les termes de la licence :

Creative Commons Attribution - Partage dans les Mêmes Conditions 3.0 France (CC BY-SA 3.0 FR) : <http://creativecommons.org/licenses/by-sa/3.0/fr/>.

Vous êtes libres :

- de **reproduire, distribuer et communiquer** cette création au public ;
- de **modifier** cette création.

Selon les conditions suivantes :

- **Attribution** : vous devez citer le nom de l'auteur original de la manière indiquée par l'auteur de l'œuvre ou le titulaire des droits qui vous confère cette autorisation (mais pas d'une manière qui suggérerait qu'ils vous soutiennent ou approuvent votre utilisation de l'œuvre) ;
- **Partage des Conditions Initiales à l'Identique** : si vous modifiez, transformez ou adaptez cette création, vous n'avez le droit de distribuer la création qui en résulte que sous un contrat identique à celui-ci.

À chaque réutilisation ou distribution de cette création, vous devez faire apparaître clairement au public les conditions contractuelles de sa mise à disposition. La meilleure manière de les indiquer est un lien vers cette page web.

Chacune de ces conditions peut être levée si vous obtenez l'autorisation du titulaire des droits sur cette œuvre.

Rien dans ce contrat ne diminue ou ne restreint le droit moral de l'auteur ou des auteurs.

Cette documentation est basée sur une réalisation du Pôle national de compétences Logiciels Libres. Les documents d'origines sont disponibles sur le site.

EOLE est un projet libre (Licence GPL).

Il est développé par le Pôle national de compétences Logiciels Libres du ministère de l'Éducation nationale, rattaché à la Direction des Systèmes d'Information de l'académie de Dijon (DSI).

Pour toute information concernant ce projet vous pouvez nous joindre :

- Par courrier électronique : eole@ac-dijon.fr
- Par FAX : 03-80-44-88-10
- Par courrier : EOLE-DSI - 2G, rue du Général Delaborde - 21000 DIJON
- Le site du Pôle national de compétences Logiciels Libres : <http://eole.orion.education.fr>

Table des matières

Chapitre 1 - Introduction	4
Chapitre 2 - Répertoires utilisés par EOLE	5
Chapitre 3 - Création de patch Creole	6
Chapitre 4 - Les dictionnaires Creole	8
1. Ajouter un en-tête XML	8
2. Utiliser des fichiers templates, paquets, services et règles de pare-feu	9
3. Utiliser des familles, variables et des séparateurs	17
4. Comportement des variables	22
5. Mettre en place des contraintes	22
6. Afficher de l'aide	29
Chapitre 5 - Le langage de template Creole	30
1. Déclarations du langage Creole	30
2. Fonctions prédéfinies	34
3. Utilisation avancée	38
4. Exemple	40
Chapitre 6 - Les scripts Creole	41
1. CreoleLint et CreoleCat	41
2. CreoleGet et CreoleSet	43
3. CreoleRun et CreoleService	45
4. CreoleLock	46
5. Indications pour la programmation	47
Chapitre 7 - Ajout de script exécuté à l'instance ou au reconfigure	51
Chapitre 8 - Ajout d'un test diagnose	53
Chapitre 9 - Gestion des noyaux Linux	55
Chapitre 10 - Gestion des tâches planifiées eole-schedule	57
Chapitre 11 - Gestion du pare-feu eole-firewall	61
Glossaire	63

Chapitre 1

Introduction

Creole^[p.63] est un ensemble d'outils permettant de mettre en œuvre un serveur suivant une configuration définie.

Il offre des possibilités de personnalisation, permettant à l'utilisateur d'ajouter des fonctionnalités sur le serveur sans risquer de créer une incohérence avec la configuration par défaut et qui ne seront pas écrasées par les futures mises à jour.

Pour personnaliser un serveur, les outils suivants sont à disposition :

- le **patch**^[p.64] : permet de modifier un **template**^[p.64] fourni par EOLE ;
- le **dictionnaire**^[p.63] **local** permet d'ajouter des options à l'interface de configuration, d'installer de nouveaux paquets ou de gérer de nouveaux services ;
- le **template**^[p.64] reprend le fichier de configuration d'une application avec, éventuellement, une personnalisation suivant des choix de configuration.

Chapitre 2

Répertoires utilisés par EOLE

Répertoires liés au logiciel Creole

Dictionnaires

- `/usr/share/eole/creole/dicos/` : contient les dictionnaires fournis par la distribution ;
- `/usr/share/eole/creole/dicos/local/` : contient les dictionnaires créés localement pour le serveur ;
- `/usr/share/eole/creole/dicos/variante/` : contient les dictionnaires fournis par une variante Zéphir.

Templates

- `/usr/share/eole/creole/distrib/` : contient tous les templates (distribution, locaux et issus de variantes) ;
- `/usr/share/eole/creole/modif/` : répertoire à utiliser pour créer des patch avec l'outil `gen_patch` ;
- `/usr/share/eole/creole/patch/` : contient les patch réalisés localement (avec ou sans l'outil `gen_patch`) ;
- `/usr/share/eole/creole/patch/variante/` : contient les patch fournis par une variante Zéphir ;
- `/var/lib/eole/` : répertoire recommandé pour le stockage des fichiers templatisés nécessitant un traitement ultérieur ;
- `/var/lib/creole/` : contient la copie des templates après la phase de patch (traitement interne à Creole).

Autres répertoires spécifiques

- `/etc/eole/` : contient les fichiers de configuration majeurs du module ;
- `/var/lib/eole/config/` : contient les fichiers de configuration de certains outils internes ;
- `/var/lib/eole/reports/` : contient des fichiers de rapport (pour affichage dans l'EAD, par exemple) ;
- `/usr/lib/eole/` : bibliothèques shell EOLE (remplacent *FonctionsEoleNg*) ;
- `/usr/share/eole/sbin/` : scripts EOLE ;
- `/usr/share/eole/diagnose/` : scripts *diagnose*.

Chapitre 3

Création de patch Creole

Si le fait de renseigner correctement les options de configuration n'offre pas une souplesse suffisante, il faut envisager des adaptations complémentaires.

Les modules EOLE sont livrés avec un ensemble de templates de fichiers de configuration qui seront copiés vers leur emplacement de destination à chaque `instance/reconfigure`.

Il est possible de personnaliser ces fichiers de configuration à l'aide d'un patch.

L'outil `gen_patch` vous permet de générer facilement un nouveau patch. Pour ce faire il suffit de copier le fichier de configuration depuis `/usr/share/eole/creole/distrib/` vers `/usr/share/eole/creole/modif/`, de le modifier et de lancer la commande `gen_patch`.



Copie du fichier du template d'origine :

```
root@scribe:~# cp /usr/share/eole/creole/distrib/php.ini
/usr/share/eole/creole/modif/
```

Changement des paramètres :

```
root@scribe:~# vim /usr/share/eole/creole/modif/php.ini
```

Exécution de la commande `gen_patch` :

```
root@scribe:~# gen_patch
** Génération des patches à partir de modif **
Génération du patch php.ini.patch
** Fin de la génération des patch **
root@scribe:~#
```

Une fois le patch créé, il faut lancer la commande `reconfigure` pour que les nouvelles options soient prises en compte.

La commande `diagnose` renvoie un diagnostic sur les patch :

```
[...]
*** Patches
. patches => Ok
[...]
```



Sont concernés par la procédure de patch uniquement les fichiers déjà présents dans le répertoire des templates et référencés dans les dictionnaires fournis par l'équipe EOLE.

Pour les autres fichiers, l'utilisation de dictionnaires locaux et de templates personnalisés est recommandée.

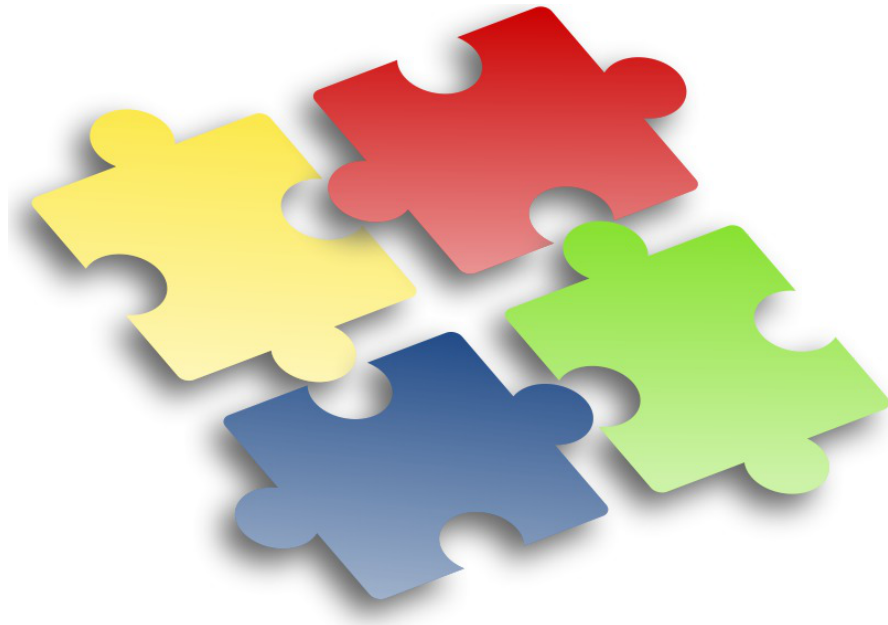
Le répertoire `/usr/share/eole/creole/` contient les répertoires suivants :

- **./distrib/** : templates originaux fournis principalement par le paquet conf d'un module ;
- **./modif/** : endroit où doivent être copiés et modifiés les templates souhaités ;
- **./patch/** : fichiers patch générés à partir des différences entre les deux répertoires précédents.

Le répertoire `/var/lib/creole/` comprend les templates finaux, c'est à dire les templates initiaux avec éventuellement des patches.



Pour désactiver un patch, il suffit de supprimer ou de déplacer le fichier patch.



Les adaptations que vous pouvez réaliser sur l'un de vos serveurs EOLE sont susceptible d'intéresser d'autres utilisateurs. Elles peuvent faire l'objet d'une intégration dans le projet EOLE par l'équipe de développement.

Les avantages sont multiples :

- pérennité de vos modifications ;
- diffusion sur l'ensemble de vos serveurs ;
- optimisé par l'équipe ;
- diffuser à tous les utilisateurs.

Aussi n'hésitez pas à proposer votre travail. Pour se faire vous pouvez vous référer à la documentation pour apprendre comment contribuer.

Chapitre 4

Les dictionnaires Creole

En cas d'ajout de templates^[p.64] et de variables supplémentaires, il est nécessaire de créer un dictionnaire local.

Ce dictionnaire peut également comprendre des noms de paquet supplémentaire à installer ainsi que des services à gérer.

Un dictionnaire local est un dictionnaire personnalisé permettant d'ajouter des options à Creole.

Un dictionnaire Creole contient un en-tête XML suivi d'une balise racine `<creole></creole>`.

Structure générale d'un dictionnaire XML Creole

```
<?xml version='1.0' encoding='utf-8'?>
<creole>
  <files>
</files>
  <containers>
</containers>
  <variables>
</variables>
  <constraints>
</constraints>
  <help>
</help>
</creole>
```



Il est toujours intéressant de regarder dans les dictionnaires déjà présents sur le module pour comprendre les subtilités des dictionnaires Creole.



Vous pouvez également vous référer à la DTD^[p.63] :

<https://dev-eole.ac-dijon.fr/projects/creole/repository/revisions/master/entry/data/creole.dtd>

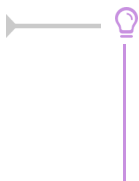
1. Ajouter un en-tête XML

L'en-tête est standard pour tous les fichiers XML :

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


Cet en-tête est nécessaire pour que le fichier soit reconnu comme étant au format XML.

Afin d'éviter les problème d'encodage, il est conseillé de créer le fichier sur un module EOLE (avec l'éditeur de texte vim).



Ajouter la configuration suivante en bas de votre fichier pour forcer l'indentation :

```
<!-- vim: ts=4 sw=4 expandtab  
-->
```

Voir aussi...

L'éditeur de texte Vim

2. Utiliser des fichiers templates, paquets, services et règles de pare-feu

Maître ou conteneur : <files> ou <containers>

Creole propose un système de conteneurs permettant d'isoler certains services du reste du système.

C'est dans le dictionnaire que les conteneurs sont définis et associés à des services.

Si le conteneur n'est pas spécifié, les services seront installés sur le serveur hôte, le maître.

Pour distinguer les fichiers templates, les paquets et les services de l'hôte de ceux mis dans le conteneur, il faut utiliser deux balises différentes.

Sur le serveur hôte, les fichiers templates, les paquets et les services sont dans une balise <files>.

Dans le cas des conteneurs, il faut spécifier un ensemble de balises <container> à l'intérieur d'une balise <containers>. L'utilisation de la balise <all> permet d'appliquer des balises à tous les <container>. En mode non conteneur cette balise s'applique sur le maître. Pour inhiber ce comportement il faut rajouter l'attribut **instance_mode='when_container'**.

La balise <container> doit obligatoirement contenir l'attribut **name** pour renseigner le nom du conteneur.

Lors de la première déclaration d'un conteneur l'attribution d'un identifiant unique (attribut **id**) est obligatoire.

La valeur de cet identifiant permettra de calculer l'adresse IP du conteneur.

Les groupes de conteneurs permettent de réunir des services afin de limiter le nombre de conteneurs.

Ils se déclarent de la même manière que les autres conteneurs. L'affectation d'un conteneur existant à un groupe de conteneurs s'effectue en utilisant l'attribut **group**.

Les ID de groupes de conteneurs de 50 à 99 sont réservés pour les groupes de conteneurs EOLE.

ID	Nom du groupe conteneur	Conteneurs inclus (AmonEcole/Eclair)
50	bdd	annuaire, mysql
51	reseau	web, mail
52	partage	fichier, dhcp, ftp
53	internet	proxy, dns
54	ltspserver	dhcp, ltsp
55	ltspapps	application

Les identifiants de conteneur supérieurs à 100 sont utilisables par les contributeurs.

La liste des identifiants des conteneurs et des groupes de conteneurs déjà affectés est actuellement maintenue sur le wiki EOLE à l'adresse :

<http://dev-eole.ac-dijon.fr/projects/creole/wiki/ContainersID>

```

1 <creole>
2   <files>
3   </files>
4   <containers>
5     <all>
6       <host hostlist='web' name='web_url' ip='adresse_ip_br0'
instance_mode='when_container' comment="Serveur web sur l'IP eth0" />
7       <file filename='/etc/fichier_cible' instance_mode=
'when_container' />
8     </all>
9     <container name='web' id='15'>
10      [...]
11    </container>
12    <container name='reseau' id='51' />
13    <!-- affectation du conteneur web au groupe de conteneurs reseau
-->
14    <container name='web' group='reseau' />
15  </containers>
16  [...]
```

Paquets : <package>

Creole permet de spécifier les paquets à installer pour profiter d'un nouveau service.

A l'instanciation de la machine, les paquets spécifiés seront installés.

Pour cela, il faut utiliser la balise <package> avec comme contenu le nom du paquet.

Les attributs de la balise <package>

- l'attribut **instance_mode** permet de définir un comportement en fonction de la présence du mode conteneur ou non : *when_container*, *when_no_container*, *always* (par défaut).



Pour spécifier plusieurs paquets, il faut obligatoirement écrire une balise <package> par

| paquet.

Fichiers templates : <file>

Les fichiers templates sont définis dans la balise <file>.

Les attributs de la balise <file>

- l'attribut **name** (obligatoire) indique l'emplacement où sera copié le fichier ;
- l'attribut **source** permet d'indiquer un nom de fichier source différent de celui de destination ;
- l'attribut **mode** permet de spécifier des droits à appliquer au fichier de destination ;
- l'attribut **owner** permet de forcer le propriétaire du fichier ;
- l'attribut **group** permet de forcer le groupe propriétaire du fichier ;
- l'attribut **filelist** permet de conditionner la génération du fichier suivant des contraintes ;
- si l'attribut **rm** vaut *True*, le fichier de destination sera supprimé si il est désactivé via une *filelist* ;
- si l'attribut **mkdir** vaut *True*, le répertoire destination sera créé si il n'existe pas ;
- l'attribut **instance_mode** permet de définir un comportement en fonction de la présence du mode conteneur ou non : *when_container*, *when_no_container*, *always* (par défaut) ;
- l'attribut **del_comment** engendre la suppression des lignes vides et des commentaires dans le fichier cible afin d'optimiser sa templatisation (exemple : `del_comment='#'`).

Renommage d'un template

L'attribut **name** contient toujours le chemin complet du fichier de destination (par exemple `/etc/hosts`).

Par défaut, le fichier template doit s'appeler de la même façon que le fichier de destination (ici : `hosts`).

Si deux templates ont le même nom, il faudra spécifier le nom du template renommé avec l'attribut **source**.

Services : <service>

Les dictionnaires Creole intègrent un système de gestion de services GNU/Linux (scripts d'init) qu'il est possible d'utiliser pour activer/désactiver des services non gérés par le module EOLE installé.

Services non gérés : services non référencés dans le système de gestion des services de Creole. Ils ne sont jamais modifiés. Ils restent dans l'état dans lequel Ubuntu les a installés ou dans celui que leur a donné l'utilisateur. Les services non gérés sont généralement les services de base Ubuntu (`rc.local`, `gpm`, ...) et tous ceux pour lesquels le module ne fournit pas de configuration spécifique (`mdadm`, ...).

Services désactivés : services systématiquement arrêtés et désactivés lors des phases d'instance et de reconfigure. Les services concernés sont généralement liés à une réponse à "non" dans l'interface de configuration du module.

Services activés : services systématiquement activés et (re)démarrés lors des phases d'instance et de reconfigure. Les services concernés sont ceux nécessaires au fonctionnement du module.

Les services à activer/désactiver se définissent dans le dictionnaire grâce à la balise <service>.

Les attributs de la balise <service>

- l'attribut **startlevel** (entier) permet de spécifier le niveau de démarrage ;
- l'attribut **stoplevel** (entier) permet de spécifier le niveau d'arrêt ;
- l'attribut **servicelist** (chaîne de caractères alphanumériques) permet de conditionner le démarrage ou l'arrêt d'un service suivant des contraintes ;
- l'attribut **method** permet de définir la façon de gérer le service : `initd`, `upstart` ou `service` (par défaut) ;
- l'attribut **hidden** (booléen) indique si le service doit être activé ou non, cet attribut est particulièrement utile lors de la redéfinition d'un service, en particulier pour forcer sa désactivation ;
- si l'attribut **pty** vaut `False`, le pseudo-terminal ne sera pas utilisé (nécessaire pour certains services) ;
- si l'attribut **redefine** vaut `True`, cela permet de redéfinir un service déjà défini dans un autre dictionnaire ;
- l'attribut **instance_mode** permet de définir un comportement en fonction de la présence ou non du mode conteneur : `when_container`, `when_no_container`, `always` (par défaut).

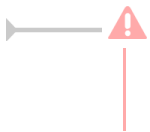
La balise `service` peut également être utilisée pour activer/désactiver des configurations de site web apache2 (commandes : `a2ensite` / `a2dissite`).

Comme pour les services système, l'activation d'un site peut être conditionnée par une `servicelist`.

On peut ainsi gérer le lien symbolique suivant : `/etc/apache2/sites-enabled/monsite` avec :

```
<service method='apache' servicelist='siteperso'>monsite</service>
```

Le fichier de configuration `monsite` étant stocké dans `/etc/apache2/sites-available/`.



Pour spécifier plusieurs services, il faut obligatoirement écrire une balise `<service>` par service.



Une règle `eole-firewall` peut être liée à un service, ainsi quand un service sera désactivé la règle le sera également.

Hôtes : <host>

La balise `<host>` permet de déclarer des hôtes à ajouter dans le fichier `/etc/hosts` du maître et/ou des conteneurs.

Les attributs de la balise <host>

- l'attribut **name** contient le nom d'une variable contenant des noms d'hôtes (FQDN), simple ou multi, obligatoire ;
- l'attribut **ip** contient le nom d'une variable contenant les adresses IPs associées aux noms, obligatoire ;
- l'attribut **hostlist** permet d'exclure cette entrée suivant des contraintes, optionnel ;
- l'attribut **crossed** combine toutes les adresses avec tous les noms d'hôtes. L'utilisation de `False` génère une association 1 nom d'hôte/1 adresse IP. Doit être `False` dans le cas d'utilisation de variables ayant une relation maître/esclave, `False`, `True` (par défaut) ;

- l'attribut **instance_mode** permet de définir un comportement en fonction de la présence du mode conteneur ou non : `when_container`, `when_no_container`, `always` (par défaut) ;
- l'attribut **comment** permet l'ajout d'une ligne de commentaire avant la(les) entrée(s), optionnel.

```

<containers>
<container name="proxy" id='20'>
<package>eole-proxy-pkg</package>
<service startlevel='30' stoplevel='30'>squid3</service>
<host hostlist='auth smb' name='nom_serveur_smb'
ip='ip_serveur_smb' instance_mode='when_container' crossed='False'
comment='serveurs d'authentification SMB' />
</container>
</containers>

```

Montage d'une partition <disknod>

La balise **<disknod>** permet de le montage d'une partition du maître à l'intérieur d'un conteneur. Par exemple, le montage de la partition `/home` dans le conteneur fichier.

Les attributs de la balise <disknod>

La balise **<disknod>** ne possède pas d'attribut spécifique.

```

<containers>
<container name='fichier' id='12'>
<disknod>/home</disknod>
</container>
</containers>

```

⚠ Pour être pris en compte il faut nécessairement arrêter le conteneur avec la commande `CreoleService lxc stop` avant de faire un `gen_conteneurs`.

Montage d'un répertoire <fstab>

La balise **<fstab>** sert à déclarer le montage d'un répertoire (qui n'est pas une partition) à l'intérieur d'un conteneur.

Par exemple, le montage du répertoire `/home/mail/` du maître dans le conteneur mail.

Les attributs de la balise <fstab>

- l'attribut **name** contient le chemin du répertoire à monter ou le nom d'une variable fournissant cette information ;

- si l'attribut **name_type** vaut *SymLinkOption* cela indique que le chemin sera défini dans la variable indiquée dans l'attribut **name** ;
- l'attribut **fstablist** (chaîne de caractères alphanumériques) permet de conditionner le montage du répertoire suivant des contraintes.

```
<containers>
  <container name='mail' id='13'>
    <fstab name='/home/mail' />
  </container>
</containers>
```

⚠ Pour être pris en compte il faut nécessairement arrêter le conteneur avec la commande `CreoleService lxc stop` avant de faire un `gen_conteneurs`.

Autorisations pour le pare-feu eole-firewall : <service_access> et <service_restriction>

`eole-firewall` est conçu pour gérer les flux réseau d'un module EOLE.

Il permet d'autoriser des connexions :

- de l'extérieur vers le maître ;
- de l'extérieur vers un conteneur.

Techniquement, ces autorisations se traduisent par des règles *iptables* et, si nécessaire, des connexions TCP Wrapper^[p.64] et l'activation de modules noyau.

⚠ eole-firewall et ERA

Pour les modules Amon et AmonEcole, les règles d'`eole-firewall` ne s'appliquent pas. Seules les règles ERA du modèle choisi s'appliquent.

Les doublons

S'il y a plusieurs règles sur une interface/port, c'est la dernière règle qui est appliquée .

Par exemple, dans le dictionnaire `20_apache.xml`, on redirige le port `80` dans le conteneur mais dans `25_nginx.xml`, on ouvre le port `80`. Si ces deux dictionnaires sont installés simultanément, c'est l'ouverture du port qui est appliquée.

L'activation des règles

Si le nom du service correspond a un service déclaré dans le conteneur et que celui-ci est désactivé, alors les accès/restrictions ne s'appliquent pas.

Si `ip` est une variable et que cette variable n'existe pas ou qu'elle est désactivée, la règle ne s'applique pas.

De la même façon pour un port/tcpwrapper avec une variable qui n'existe pas, aucune règle ne s'applique.

Malgré son nom, l'attribut `service` des balises `service_access` et `service_restriction` doit être renseigné avec le nom de la `servicelist` associée au service et non avec le nom du service lui-même.

Si aucune `servicelist` permettant de désactiver le service n'existe, l'attribut peut être rempli librement.

Autoriser un port (XXX) pour un service donné (YYY) :

```
<service_access service='YYY'>
  <port>XXX</port>
</service_access>
```

Dans la balise `port` il est également possible de spécifier le protocole (par défaut c'est TCP).

Par exemple :

```
<service_access service='ntp'>
  <port protocol='udp'>123</port>
</service_access>
```

Avec `tcpwrapper` :

```
<tcpwrapper>YYY</tcpwrapper>
```

Port avec variable (ZZZ) :

```
<port port type="SymLinkOption">ZZZ</port>
```

List (WWW) pour port/tcpwrapper :

```
<port service_accesslist="WWW">XXX</port>
<tcpwrapper service_accesslist="WWW">YYY</tcpwrapper>
```

⦿ Règles `eole-firewall` extraites du dictionnaire `/usr/share/eole/creole/dicos/01_network.xml` pour le service `sshd`

```
1 <service_access service='sshd'>
2   <port>22</port>
3   <tcpwrapper>sshd</tcpwrapper>
4 </service_access>
5 <service_restriction service='sshd'>
6   <ip interface='eth0' netmask='netmask_ssh_eth0' netmask_type=
7   'SymLinkOption' ip_type='SymLinkOption'>ip_ssh_eth0</ip>
7   <ip interface='eth1' netmask='netmask_ssh_eth1' netmask_type=
8   'SymLinkOption' ip_type='SymLinkOption'>ip_ssh_eth1</ip>
8   <ip interface='eth2' netmask='netmask_ssh_eth2' netmask_type=
9   'SymLinkOption' ip_type='SymLinkOption'>ip_ssh_eth2</ip>
9   <ip interface='eth3' netmask='netmask_ssh_eth3' netmask_type=
10  'SymLinkOption' ip_type='SymLinkOption'>ip_ssh_eth3</ip>
10  <ip interface='eth4' netmask='netmask_ssh_eth4' netmask_type=
11  'SymLinkOption' ip_type='SymLinkOption'>ip_ssh_eth4</ip>
11 </service_restriction>
```

Si on ne définit que les `service_access`, le port est ouvert pour tout le monde sur toutes les interfaces.

Pour ajouter des restrictions il faut mettre :

```
<service_restriction service='YYY'>
  <ip interface='eth0'>1.1.1.1</ip>
</service_restriction>
```

Dans ce cas, seule l'adresse IP 1.1.1.1 peut accéder à ce service.

Il est possible d'utiliser des variables :

```
<ip interface='auto' ip_type='SymLinkOption'>variable</ip>
```

Il est possible d'utiliser un netmask :

```
<ip interface='eth0' netmask="255.255.255.0"
ip_type='SymLinkOption'>variable</ip>
<ip interface='eth1' netmask="variable_netmask"
netmask_type='SymlinkOption' ip_type='SymLinkOption'>variable</ip>
```

Le paramètre interface peut être :

- ethX (pour une interface donnée) ;
- all (pour toutes les interfaces) ;
- auto (calcul de l'interface via la route) ;
- une variable (avec l'ajout de interface type="SymlinkOption").

Il est aussi possible d'ajouter une service_restrictionlist aux balises ip.

🔍 Règles eole-firewall extraites du dictionnaire

/usr/share/eole/creole/dicos/01_network.xml pour le service genconfig

```
1 <service_access service='genconfig'>
2   <port>7000</port>
3 </service_access>
4 <service_restriction service='genconfig'>
5   <ip interface='eth0' netmask='netmask_ssh_eth0' netmask_type=
6     'SymLinkOption' ip_type='SymLinkOption'>ip_ssh_eth0</ip>
7   <ip interface='eth1' netmask='netmask_ssh_eth1' netmask_type=
8     'SymLinkOption' ip_type='SymLinkOption'>ip_ssh_eth1</ip>
9   <ip interface='eth2' netmask='netmask_ssh_eth2' netmask_type=
10    'SymLinkOption' ip_type='SymLinkOption'>ip_ssh_eth2</ip>
11   <ip interface='eth3' netmask='netmask_ssh_eth3' netmask_type=
12    'SymLinkOption' ip_type='SymLinkOption'>ip_ssh_eth3</ip>
13   <ip interface='eth4' netmask='netmask_ssh_eth4' netmask_type=
14    'SymLinkOption' ip_type='SymLinkOption'>ip_ssh_eth4</ip>
15 </service_restriction>
```

Complément sur les attributs

instance_mode

L'attribut instance_mode remplace les anciens attributs in_container et container_only.

Une ressource, qu'elle soit sur le maître ou dans un conteneur, peut n'être à générer que si le mode conteneur est activé ou désactivé :

instance_mode	mode conteneur	mode non conteneur
when_container	✓	
when_no_container		✓
always (default)	✓	✓

Les balises acceptant l'attribut instance_mode sont actuellement :

- package ;
- file ;
- service ;
- host.

Exemple récapitulatif

Fichiers templates, paquets et services locaux ou dans un conteneur

```

1 <containers>
2   <!-- dans le conteneur mon_reverseproxy -->
3   <container name="mon_reverseproxy" id='101'>
4     <package>nginx</package>
5     <service servicelist="myrevprox" startlevel='91'>nginx</service>
6     <file filelist='myrevprox' name='/etc/nginx/sites-enabled/default'
7       source='nginx.default' />
8     <file filelist='myrevprox' name='/var/www/nginx-default/nginx.html' rm
9       = 'True' />
10    </container>
11  </containers>
12  <files>
13    <!-- sur le maître-->
14    <service>ntp</service>
15    <file name='/etc/ntp.conf' />
16    <file name='/etc/default/ntpdate' owner='ntp' group='ntp' mode='600' />
17    <file name='/etc/strange/host' source='strangehost.conf' mkdir='True' />
18  </files>

```

Voir aussi...

Choisir le mode du module

3. Utiliser des familles, variables et des séparateurs

Variables : <variables>

L'ensemble des familles et, ainsi, des variables sont définies dans un nœud <variables></variables>.

Familles : <family>

Un conteneur famille permet d'avoir des catégories de variables. Celle-ci correspond à un onglet dans l'interface. Les familles sont incluses obligatoirement dans une balise <variables>.



Une famille `Squid` pour gérer toutes les variables relatives a *Squid*.

Les attributs de la balise *family* sont les suivants :

- l'attribut **name** (obligatoire) est à la fois le nom et l'identifiant de la famille ;
- l'attribut **mode** permet de définir le mode d'affichage de la famille :
 - mode basic par défaut ;
 - mode normal ;
 - mode expert.
- l'attribut **icon** définit une image associée à l'onglet ;
- l'attribut **hidden** indique si la famille doit être affichée ou non, sa valeur pouvant être modifiée via une condition (voir plus bas).



Une famille dont toutes les variables sont cachées (hidden) ou désactivées (disabled) ne sera pas affichée sauf en mode debug.



Les icônes utilisés proviennent des bibliothèques de polices et d'icônes libres :

- Font Awesome : <http://fontawesome.github.io/Font-Awesome/icons> ;
- Font Mfizz : <http://fizzed.com/oss/font-mfizz>.

Pour choisir une icône, il faut se rendre sur les pages ci-dessus et recopier le nom de l'icône. Pour la font Mfizz il faut enlever le préfixe `icon-`.



```
<family name='messagerie' mode='basic' icon='envelope'>
<variable name='system mail from' type='mail' description="Adresse
électronique d'envoi pour le compte root"/>
</family>
```

Variable : <variable>

Une variable contient une description et, optionnellement, une valeur EOLE par défaut.

Les variables peuvent être à valeur unique ou multi-valuées.

Les balises **<variable>** sont incluses obligatoirement dans une balise **<family>**.

Les attributs de la balise *variable* sont les suivants :

- l'attribut **name** (obligatoire) est le nom de la variable ;
- l'attribut **type** (obligatoire) permet d'utiliser un type EOLE avec des vérifications automatiques (fonctions de vérifications associées a chaque type de variable) ;
- l'attribut **description** permet de définir le libellé à afficher dans l'interface de configuration du module ;

- l'attribut **multi** permet de spécifier qu'une variable pourra avoir plusieurs valeurs (par exemple pour un DNS, on aura plusieurs adresses IP de serveurs DNS) ;
- l'attribut **mode** permet de définir le mode d'affichage de la variable (*basic*, *normal* ou *expert*) ;
- si l'attribut **hidden** vaut *True*, la variable ne sera pas affichée dans l'interface de configuration (on peut par exemple souhaiter masquer des variables dont la valeur est calculée automatiquement) ;
- si l'attribut **disabled** vaut *True*, la variable sera déclarée comme désactivée.
- si l'attribut **mandatory** vaut *True*, la variable sera considérée comme obligatoire, cet attribut remplace l'ajout d'un *check obligatoire* au niveau des conditions ;
- si l'attribut **redefine** vaut *True*, cela permet de redéfinir une variable déjà définie dans un autre dictionnaire ;
- si l'attribut **exists** vaut *False*, cela permet de définir une variable si et seulement si elle n'a pas déjà été définie dans un autre dictionnaire.
- si l'attribut **remove_check** vaut *True* pour une variable redéfinie, alors toutes les validations (*check*) associées à cette variable sont réinitialisées ;
- si l'attribut **remove_condition** vaut *True* pour une variable redéfinie, alors toutes les conditions associées à cette variable sont réinitialisées (attention, **remove_condition** n'est disponible qu'à partir de la version **EOLE 2.5.2**) ;
- si l'attribut **auto_freeze** vaut *True*, la variable devient à verrouillage automatique. Sa valeur est verrouillée dès le premier enregistrement de la configuration. Dans l'interface de configuration du module, ces variables sont identifiées par la présence d'un cadenas. Ce dernier apparaît verrouillé une fois le serveur instancié ;
- si l'attribut **auto_save** vaut *True*, la variable devient à enregistrement obligatoire. Sa valeur est obligatoirement enregistrée dans le fichier de configuration et elle n'est donc pas automatiquement modifiée si sa valeur par défaut change au niveau des dictionnaires. On retrouve ainsi un fonctionnement équivalent à celui disponible sur EOLE 2.3.

Les principaux types de variables Creole sont les suivants :

- *number* : la valeur de la variable doit être du type "int". La fonction python `int(value)` ne doit pas retourner d'erreur ;
- *string* : la valeur de la variable doit être du type "unicode" ;
- *ip* : valeur de type IP. La valeur doit passer ce test : `IPy.IP('{0}/32'.format(value))` ;
- *local_ip* : la même chose que IP, sauf que les adresses réservées et privées soulèvent un warning (voir *IPy* pour des informations sur les adresses réservées et privées) ;
- *netmask* : adresse de masque réseau. La valeur doit passer ce test : `IPy.IP('0.0.0.0/{0}'.format(value))` ;
- *network* : adresse réseau. La valeur doit passer ce test : `IPy.IP(value)` ;
- *broadcast* : adresse de broadcast. : La valeur doit passer ce test : `IPy.IP('{0}/32'.format(value))` ;
- *netbios* : alphanumérique autorisé sauf pour le 1er caractère qui doit forcément être du type alpha, minimum 2 et maximum 15 caractères ;
- *domain* :

- adresse IP. La valeur doit passer ce test : `IPy.IP('{0}/32'.format(value))`

ou

- alphanumérique et '.' autorisé sauf pour le 1er caractère qui doit forcément être du type alpha. Le '.' est obligatoire. Minimum 2 et maximum 255 caractères ;
- *domain_strict* : nom DNS uniquement (adresse IP interdite) ;
- *unix_user* : nom d'utilisateur ou de groupe Unix ;
- *web_address* : adresse Internet. Doit débuter par `http://` ou `https://` ;
- *hostname* :
 - adresse IP. La valeur doit passer ce test : `IPy.IP('{0}/32'.format(value))`

ou

- alphanumérique autorisé sauf pour le 1er caractère qui doit forcément être du type alpha. Minimum 2 et maximum 63 caractères ;
- *hostname_strict* : nom d'hôte uniquement (adresse IP interdite) ;
- *mail* : adresse e-mail ;
- *port* : entier compris entre 1 et 65535 ;
- *filename* : tout chemin Unix (fichier ou répertoire) ;
- *oui/non* : seules valeurs possibles : "oui" et "non" ;
- *yes/no* : seules valeurs possibles : "yes" et "no" ;
- *on/off* : seules valeurs possibles : "on" et "off" ;

Comportement avec `redefine=True` et `remove_check=False`

- si la nouvelle variable fournit une valeur par défaut, elle remplace l'ancienne ;
- si la nouvelle variable fournit un ou plusieurs des attributs suivants : *description*, *hidden*, *mandatory*, *auto_freeze*, *mode*, les valeurs des nouveaux attributs remplacent les anciennes ;
- les attributs *type* et *multi* ne sont pas modifiables ;
- si un nouveau *valid_enum* est défini dans les fonctions *checks*, il remplace l'ancien ;
- si de nouveaux *disabled_if(_not)_in* sont définis, ils remplacent les anciens ;
- les autres conditions et contraintes sont ajoutées à celles qui étaient déjà définies.

Valeur : <value>

A l'intérieur d'une balise <variable>, il est possible de définir une balise <value> permettant de spécifier la valeur par défaut de la variable.

Séparateurs : <separators> et <separator>

Les séparateurs permettent de définir des barres de séparation au sein d'une famille de variable dans l'interface de configuration.

Les séparateurs définis dans un dictionnaire sont placés dans la balise <separators></separators> dans la balise <variables>.

A l'intérieur de la balise **<separators>** il faut spécifier autant de balises **<separator>** que de séparateurs souhaités.

Les attributs de la balise *separator* sont les suivants :

- l'attribut **name** (obligatoire) correspond au nom de la variable suivant le séparateur ;
- si l'attribut **never_hidden** vaut *True*, le séparateur sera affiché même si la variable associée est masquée.

Dans le cas où il n'y a aucune variable à afficher dans le bloc défini par le séparateur, celui-ci est forcément masqué.

Exemple

Variables, familles et séparateurs

```
<variables>
  <family name='services'>
    .. <variable name='activer_esu' type='oui/non'
description="Utiliser le logiciel ESU" hidden='True'>
    .. <value>oui</value>
    .. </variable>
  .. </family>
  .. <family name='esu'>
    .. <variable name='esu_proxy' type='oui/non'
description="Activer le proxy ESU">
    .. <value>non</value>
    .. </variable>
    .. <variable name='esu_proxy_server' type='domain'
description='Adresse du proxy ESU' mandatory='True' />
    .. <variable name='esu_proxy_port' type='port' description='Port
du proxy ESU' mandatory='True'>
    .. <value>3128</value>
    .. </variable>
    .. <variable name='esu_proxy_bypass' type='string'
description='Ne pas utiliser le proxy ESU pour' multi='True'>
    .. <value>127.0.0.1</value>
    .. </variable>
  .. </family>
  .. <separators>
    .. <separator name='esu_proxy'>Proxy ESU</separator>
  .. </separators>
</variables>
```

4. Comportement des variables

En plus des propriétés décrites explicitement dans les dictionnaires Creole, certaines variables se voient ajouter des contraintes ou des modifications de propriétés en fonction de certains paramètres.

Les variables possédant la propriété `auto_freeze='True'` sont obligatoirement affichées en mode basique lors de la saisie initiale, ceci afin d'attirer l'attention de l'utilisateur sur le fait qu'elles ne seront plus modifiables ultérieurement.

Une exception a été ajoutée à cette règle, si la propriété `expert='True'` est explicitement ajoutée sur la variable, celle-ci apparaîtra uniquement dans le mode expert.

Les variables obligatoires (`mandatory='True'`) ne possédant pas de valeur par défaut (calculée ou non) sont obligatoirement affichées en mode basique, puisque l'utilisateur devra forcément les renseigner.

Les variables non obligatoires (`mandatory='False'`) possédant une valeur par défaut (balise `<value>`) deviennent obligatoires.

5. Mettre en place des contraintes

Des fonctions (contraintes) peuvent être utilisées pour grouper / tester / remplir / conditionner des variables.

L'ensemble des contraintes d'un dictionnaire se place à l'intérieur d'un nœud XML `<constraints></constraints>`.

Lien entre variables : `<group>`

Il est possible de lier des variables sous la forme d'une relation maître-esclave(s).

La variable maître doit obligatoirement être multi-valuée (`multi='True'`).

Elle se définit dans l'attribut **master**.

Les variables esclaves sont définies entre les balises `<slave>`.

Les variables esclaves deviennent automatiquement multi-valuées.

```

<group master='adresse_ip_eth0'>
  <slave>adresse_netmask_eth0</slave>
  <slave>adresse_network_eth0</slave>
</group>

```

Calcul automatique modifiable `<fill>` ou non `<auto>`

Le calcul automatique permet d'associer automatiquement (par le calcul) une valeur par défaut à une variable.

Cette valeur peut être :

- éditable par l'utilisateur : balise `<fill>` ;
- non éditable par l'utilisateur (exemple : l'IP d'un serveur en DHCP) : balise `<auto>`.

⚠ Contrairement aux versions précédentes si l'utilisateur a choisi de conserver la valeur par défaut d'une variable affectée par un *fill*, le calcul de la valeur sera réalisé à chaque fois, comme pour les variables *auto* sauf si la variable possède l'attribut `auto_save='True'`.

⚠ Les calculs *auto* ne sont pas compatibles avec les variables à verrouillage automatique (`auto_freeze`) ou à enregistrement obligatoire (`auto_save`).

L'attribut *name* correspond au nom de la fonction qui sera utilisée pour le calcul.

Les fonctions utilisées peuvent être :

- des fonctions natives de Tiramisu^[p.64] ;
- des fonctions déclarées dans le fichier `eosfunc.py` ;
- des fonctions ajoutées en tant que fonctions personnalisées (voir ci-dessous).

L'attribut de la balise *param* : `optional='True'` : indique que le paramètre sera ignoré si la variable associée n'existe pas. Cela permet de contourner les erreurs du type : `Utilisation de la variable <param var name> non présente dans un calcul`

L'attribut de la balise *param* : `hidden='False'` : indique que le paramètre sera ignoré si la variable possède des propriétés incompatibles avec le calcul (variable désactivée, variable obligatoire sans valeur, ...). Cela permet de contourner les erreurs du type : `impossible d'effectuer le calcul, l'option <target var name> a les propriétés : ['disabled'] pour : <param var name>`

Les principales fonctions de calcul utilisables avec les balises *fill* et *auto* sont les suivantes :

- *calc_network* : calcule l'adresse réseau par défaut à partir d'une IP et d'un masque de sous-réseau


```
<fill name='calc_network' target='my_network'>
  <param type='eole' name='ip'>my ip</param>
  <param type='eole' name='netmask'>my_netmask</param>
</fill>
```
- *calc_broadcast* : calcule l'adresse de broadcast à partir d'une adresse IP et d'un masque de sous-réseau


```
<fillname='calc_broadcast' target='my_broadcast'>
  <param type='eole' name='ip'>my ip</param>
  <param type='eole' name='netmask'>my_netmask</param>
</fill>
```

- *calc_val* : renvoie la valeur passée en paramètre (généralement la valeur d'une autre variable)

```
<fill name='calc_val' target='nom_machine'>
  <param type='eole' name='valeur'>eole module</param>
</fill>
```

- *calc_val_first_value* : renvoie la valeur de la première variable définie

```
<fill name='calc_val_first_value' target='eolessa_adresse'>
  <param type='eole' optional='True' hidden='False'>web_url</param>
  <param type='eole'>adresse_ip_eth0</param>
</fill>
```

- *calc_multi_val* : renvoie les valeurs passées en paramètre en supprimant les doublons

```
<fill name='calc_multi_val' target='ssl_organization_unit_name'>
  <param>110 043 015</param>
  <param type='eole'>nom_academie</param>
  <param type='eole'>numero_etab</param>
</fill>
```

- *concat* : concaténation de plusieurs valeurs

```
<fill name="concat" target='bacula_dir_name'>
  <param type='eole' name='valeur1'>nom_machine</param>
  <param name='valeur2'>-dir</param>
</fill>
```

- *calc_multi_condition* : la valeur est déterminée en fonction d'une ou de plusieurs autres variables. Le résultat peut être une chaîne de caractères ou la valeur d'une autre variable multi ou non (si type='eole')

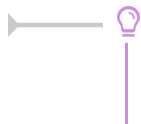
```
<auto name='calc_multi_condition' target='variable_calculée'>
  <param>oui</param>
  <param type='eole' name='condition_1'>activer_logiciel1</param>
  <param type='eole' name='condition_2'
  hidden='False'>activer_logiciel2</param>
  <param name='match'>oui</param>
  <param name='mismatch' type='eole'>variablemiss</param>
  <param
  name='default_mismatch'>valeur_si_variablemiss_disabled</param>
</auto>
```

Il est possible d'utiliser des *calc_multi_condition* avec des variables non déclarées ou désactivées mais uniquement si toutes les variables testent la même condition.

A *contrario*, il est possible de spécifier une condition différente pour chacune des variables en fournissant la liste dans la première balise param : `<param>['non', 'oui']</param>`. Dans ce cas, il faut exactement le bon nombre de variables et que celles-ci soient accessibles.

Validation et/ou liste de choix : <check>

La valeur renseignée pour une variable est validée par une fonction.



La déclaration de nombreuses validations peut être évitée en affectant un type adapté à chacune des variables.

L'attribut *name* correspond au nom de la fonction qui sera utilisée pour le calcul.

Les fonctions utilisées peuvent être :

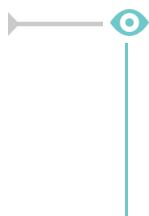
- des fonctions natives de Tiramisu^[p.64] ;
- des fonctions déclarées dans le fichier `eosfunc.py` ;
- des fonctions ajoutées en tant que fonctions personnalisées (voir ci-dessous).

L'attribut de la balise *param* : *optional='True'* : indique que le paramètre sera ignoré si la variable associée n'existe pas. Cela permet de contourner les erreurs du type : Utilisation de la variable <param var name> non présente dans un calcul

L'attribut de la balise *param* : *hidden='False'* : indique que le paramètre sera ignoré si la variable possède des propriétés incompatibles avec le calcul (variable désactivée, variable obligatoire sans valeur, ...). Cela permet de contourner les erreurs du type : impossible d'effectuer le calcul, l'option <target var name> a les propriétés : ['disabled'] pour : <param var name>

La présence de l'attribut **level="warning"** indique que le test de validation n'est pas bloquant.

En cas d'échec de la validation un message d'alerte apparaîtra mais la valeur sera tout de même acceptée.



```
<check name="valid_ipnetmask" target="adresse_netmask_eth0"
level="warning">
  <param type='eole'>adresse ip eth0</param>
</check>
```

Les principales fonctions de validation disponibles sont les suivantes :

- *valid_enum* : la valeur doit être choisie parmi celles de la liste, si *checkval* est à False, l'utilisateur est autorisé à entrer la valeur de son choix (liste ouverte)

```
<check name="valid_enum" target="lettre">
  <param>['a','b','c']</param>
  <param name="checkval">False</param>
</check>
```

- *valid_regexp* : la valeur doit être conforme à une expression rationnelle

```
<check name='valid_regexp' target='code_ent'>
  <param>^[A-Z][0-9]${</param>
  <param name='err_msg'>L'identifiant doit etre compose d'une lettre
et d'un chiffre</param>
```

```
</check>
```

- *valid_differ* : la valeur doit être différente de celle passée en paramètre

```
<check name='valid_differ' target='smb_workgroup'>
```

```
  <param type='eole' hidden='False'>smb_netbios_name</param>
```

```
</check>
```

- *valid_entier* : la valeur doit être un entier sur lequel on peut définir un minimum et/ou un maximum

```
<check name='valid_entier' target='nombre'>
```

```
  <param name='mini'>0</param>
```

```
  <param name='maxi'>50</param>
```

```
</check>
```

- *valid_networknetmask* : vérifie la cohérence entre une variable de type *network* et la variable de type *netmask* associée

```
<check name="valid_networknetmask" target="netmask_ssh_eth0">
```

```
  <param type='eole'>ip_ssh_eth0</param>
```

```
</check>
```

- *valid_ipnetmask* : vérifie la cohérence entre une variable de type *ip* et la variable de type *netmask* associée

```
<check name="valid_ipnetmask" target="adresse_netmask_eth0"
level="warning">
```

```
  <param type='eole'>adresse_ip_eth0</param>
```

```
</check>
```

- *valid_in_network* : vérifie la cohérence entre une variable de type *ip* et les variables de type *network* et *netmask* associées

```
<check name="valid_in_network" target="adresse_ip_gw">
```

```
  <param type='eole'>adresse_network_eth0</param>
```

```
  <param type='eole'>adresse_netmask_eth0</param>
```

```
</check>
```

Autre fonction de validation disponible mais non utilisée dans les dictionnaires livrés :

- *valid_broadcast*

Contrainte entre variables : <condition>

disabled_if_in et disabled_if_not_in

Les conditions *disabled_if_in* et *disabled_if_not_in* permettent :

- d'activer/désactiver une variable (*type='variable'*)
- d'activer/désactiver une famille (*type='family'*)
- d'activer/désactiver des services (*type='servicelist'*)
- d'activer/désactiver la templatisation de fichiers (*type='filelist'*)

en fonction d'un ensemble de conditions.

```

<condition name='disabled_if_not_in' source='port_rpc'>
  <param>0</param>
  <param>7080</param>
  <target>ip eth0</target>
  <target type='family' optional='True'>net</target>
  <target type='filelist'>ldap</target>
  <target type='servicelist'>ldap</target>
</condition>

```

Si l'attribut **optional** de la balise target vaut **'True'**, la cible sera ignorée si elle n'existe pas.

Cela permet de contourner les erreurs du type : Variable <target var name> inexistante mais avec condition

Si l'attribut **fallback** de la balise condition vaut **'True'**, les cibles seront automatiquement désactivées si le calcul de la condition est impossible (variable source inconnue ou désactivée).

Cela permet de contourner les erreurs du type : Variable <src var name> inexistante mais utilisée dans une condition

Son utilisation évite d'avoir à déclarer explicitement la variable source avec l'attribut *exists='False'* dans le dictionnaire courant.

```

<condition name='disabled_if_in' source='activer_spamassassin'
  fallback='True'>
  <param>non</param>
  <target type='variable'>exim spam score</target>
</condition>

```

! hidden_if_in et hidden_if_not_in

Les anciennes conditions *hidden_if_in* et *hidden_if_not_in* sont toujours supportées mais leur comportement est désormais calqué sur celui des *disabled_if_in* et *disabled_if_not_in* par lesquelles elles doivent être remplacées.

frozen_if_in et frozen_if_not_in

Les conditions *frozen_if_in* et *frozen_if_not_in* permettent de passer une variable en mode automatique (valeur non modifiable par l'utilisateur) en fonction d'un ensemble de conditions.

```

<condition name='frozen if not in' source='eth0 method'>
  <param>statique</param>
  <target type='variable'>adresse_ip_eth0</target>

```

```
<target type='variable'>adresse_netmask_eth0</target>
<target type='variable'>adresse_ip_gw</target>
</condition>
```

Ajout de fonctions personnalisées

Il est possible d'ajouter des bibliothèques de fonctions personnalisées dans le répertoire `/usr/share/creole/funcs`.

Les bibliothèques doivent posséder l'extension `.py` et contenir des fonctions python.



```
# -*- coding: utf-8 -*-
def to_iso(data):
    """ encode une chaine en ISO """
    try:
        return unicode(data, "UTF-8").encode("ISO-8859-1")
    except:
        return data
```



Si vous devez importer des bibliothèques python dans un fichier de fonctions personnalisées, ne les importez pas en début de fichier. Les imports doivent être faits dans la fonction de calcul elle-même.



Les adaptations que vous pouvez réaliser sur l'un de vos serveurs EOLE sont susceptibles d'intéresser d'autres utilisateurs. Elles peuvent faire l'objet d'une intégration dans le projet EOLE par l'équipe de développement.

Les avantages sont multiples :

- pérennité de vos modifications ;
- diffusion sur l'ensemble de vos serveurs ;
- optimisé par l'équipe ;
- diffuser à tous les utilisateurs.

Aussi n'hésitez pas à proposer votre travail. Pour se faire vous pouvez vous référer à la documentation pour apprendre comment contribuer.

6. Afficher de l'aide

Il est possible d'afficher de l'aide dans l'interface :

- affichée au survol de l'onglet : **<family>** ;
- affichée au survol du libellé de la variable : **<variable>**.

L'ensemble des aides d'un dictionnaire est dans la balise **<help>**.

```

<help>
  <variable name='adresse ip eth0'>
    Adresse IP de la premiere carte réseau (ex: 10.21.5.1)
  </variable>
</help>
<help>
  <family name='messagerie'> Paramétrage du serveur de
messagerie (MTA) Exim :
  - Paramétrage d'Exim selon 5 modèles ;
  - Paramétrage du domaine de messagerie suivant le modèle
Exim ;
  - Paramétrage des réécritures d'adresses ;
  - Paramétrage des logs Exim ;
  - Paramétrage du relais des mails ;
  - Paramétrage d'activation de spamassassin ;
  - Paramétrage d'activation de Sympa.
</family>
</help>

```

Chapitre 5

Le langage de template Creole

Les variables du dictionnaire Creole sont accessibles en les préfixant par la chaîne de caractères : `%%`.

Si dans le dictionnaire Creole :

```
adresse_ip_eth0 vaut 192.168.170.1
```

Et qu'on a dans un template source le contenu suivant :

```
bla bla bla %%adresse_ip_eth0 bla bla bla
```

Après instanciation, le fichier cible contiendra :

```
bla bla bla 192.168.170.1 bla bla bla
```



Dans les cas où une variable est susceptible d'être confondue avec le texte qui l'entoure, il est possible d'encadrer son nom par des accolades :

```
%%{adresse_ip_eth0} est identique à %%adresse_ip_eth0.
```

1. Déclarations du langage Creole

Creole fournit un langage de template complet.

Il est possible de créer des boucles, des tests, de gérer les lignes optionnelles, de réaliser des inclusions répétées, ...

La déclaration de test : if

Syntaxe :

```
%if EXPRESSION |code if %else |code else %end if
```

Dans les tests il est possible d'utiliser les opérateurs du langage python : `==`, `!=`, `>`, `<`, `>=`, `<=`, `not`, `and`, `or`, ...



```
%if %%size > 500
c'est grand
%elif %%size >= 250
c'est moyen
%else
c'est petit
%end if
```

```

%if %%toto == 'yes' and ( %%titi != "" or %%tata not in
['a','b','c'] ) :
  la condition a été validée
%end if

```

La déclaration d'itération : for

Syntaxe :

```

%for %%iterateur in EXPRESSION
CODE avec %%iterateur
%end for

```

La boucle `%%for` est particulièrement intéressante lorsque l'on souhaite effectuer des traitements sur une **variable multi-valuée**.

```

%for %%i in range(4)
  itération %%i
%end for

%for %%valeur in %%variable multivaluee
  %%valeur
%end for

```

Pour des traitements simples, la fonction prédéfinie `%%custom_join` (voir section suivante) peut avantageusement éviter la mise en place d'une boucle `%for`.

La notation pointée

Si une variable Creole est **multivaluée** et **maître** (*master d'un groupe de variable*) alors, il est possible de faire appel à ses variables **esclaves** à l'intérieur de la boucle `%for`.

Si `netmask_admin_eth0` est esclave de `ip_admin_eth0` alors, il est possible d'appeler cette variable en notation pointée.

Par exemple : dans le dictionnaire Creole figurent les variables suivantes.

`ip_admin_eth0` est la variable maître et :

- `ip_admin_eth0 = ['1.1.1.1', '2.2.2.2']`
- `netmask_admin_eth0 = ['255.255.255.255', '255.255.255.255']`

Le template suivant :

```

%for %%ip_admin in %%ip_admin_eth0
  %%ip_admin/%%ip_admin.netmask_admin_eth0
%end for

```

donnera comme résultat :

```
1.1.1.1/255.255.255.255
```

```
2.2.2.2/255.255.255.255
```

Il est également possible aussi d'accéder à l'index (la position dans la liste) de la variable en cours de boucle :

```
%for %%idx, %%val in %%enumerate(%%ip_admin_eth0)
```

```
L'index de %%val est : %%idx
```

```
%end for
```

Le template généré sera le suivant :

```
l'index de : 1.1.1.1 est : 0
```

```
l'index de : 2.2.2.2 est : 1
```

Il est également possible (mais déconseillé) d'utiliser une "notation par item" (notation entre crochets). Par exemple pour accéder à l'item numéro 5 d'une variable, il faut écrire :

```
variable[5]
```

La variable doit être évidemment être **multivaluée** et comporter au minimum (*item+1*) valeurs.

```
ip_admin_eth0 = ['1.1.1.1', '2.2.2.2', '3.3.3.3']
```

et si un template a la forme suivante :

```
bla bla
```

```
%%ip_admin_eth0[2]
```

```
bla bla
```

alors l'instanciation du template donnera comme résultat :

```
bla bla
```

```
3.3.3.3
```

```
bla bla
```


.value et .index

Les attributs *.value* et *.index* ne sont plus supportés et ne doivent plus être utilisés dans les templates.

Les déclarations spéciales echo et set

L'instruction `%echo` permet de déclarer une chaîne de caractères afin que celle-ci apparaisse telle quelle dans le fichier cible.

Cela est utile lorsqu'il y a des caractères spéciaux dans le template source et, en particulier, les caractères `%` et `\` qui sont susceptibles d'être interprétés par le système de template.

 `%echo "- deux barres obliques : \\\n- un pourcentage : %"`

L'utilisation de l'instruction `%echo` ne rend pas les templates très lisibles d'autant plus que, généralement, on souhaite intercaler des variables au milieu des caractères spéciaux.

En pratique, il est donc préférable de passer par des variables locales que l'on peut déclarer avec `%set`.

```

%set %%slash='\'
%set %%double_slash='\\\'
%%double_slash%%machine%%{slash}partage

```

Autres déclarations

La déclaration while

Syntaxe : `%while` `EXPR` `contenu`

`%end while`

Exemple :

```

%while %someCondition('arg1', %%arg2)
The condition is true.
%end while

```

La déclaration repeat

Syntaxe : `%repeat` `EXPR`

`%end repeat`

La déclaration unless

`%unless` `EXPR`

`%end unless`

peut être utile si une variable est dans le dictionnaire Creole pour "ne pas" exécuter une action :

```

%unless %%alive
do this
%end unless

```

La syntaxe d'inclusion

il est possible d'inclure des fichiers à l'aide de la déclaration suivante :

```
%include "includeFileName.txt"
```

ou bien à partir du nom long du fichier à inclure (le nom de fichier étant ici renseigné dans une variable Creole :

```
%include source=%%myParseText
```

Effacement des retours chariots : slurp

Exemple d'utilisation :

```

%for %%i in range(15)
%%i-%slurp
%end for

```

donnera :

```
1-2-3-4-5-6...
```

sur une seule ligne (gobe les retours chariots)

remarquons que dans ce cas là, `slurp` n'est pas nécessaire et il est possible d'écrire le end sans sauter de ligne :

```
%for %%i in range(15)
%%i-%end for
```

exemple 2 :

```
%if %%dns nameservers != ['']
dns nameservers %slurp
%for %%name server in %%dns nameservers %%name server %slurp
%end for
%end if
#
```

générera :

```
dns nameserver toto titi #
```

2. Fonctions prédéfinies

Il est possible d'accéder à des fonctions prédéfinies, provenant du module : `eosfunc.py`.

Ces fonctions peuvent être utilisées dans un template de la manière suivante (exemple) :

```
[...] %%fonction_predefinie(%%variable) [...]
```

Variable "optionnelle" : `is_defined`

Il peut arriver qu'on ne soit pas sûr que la variable que l'on souhaite tester soit définie dans les dictionnaires présents sur le module ou que la variable soit désactivée.

C'est le cas lorsque l'on veut traiter un cas particulier dans un template qui est commun à plusieurs modules.

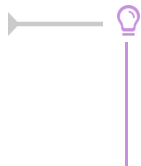
Hors, si une variable est utilisée dans le template cible sans avoir été définie, le processus d'instanciation sera stoppé.

Pour tester si une variable est définie, il faut utiliser la fonction `%%is_defined`.

```
%if %%is_defined('ma_variable')
%%ma_variable
%else
la variable n'est pas définie
%end if
```

Contrairement à toutes les autres fonctions, `is_defined` nécessite comme argument le nom de la variable fourni sous forme d'une **chaîne de caractères**.

Si une variable non définie est placée dans un bloc qui n'est pas traité (conditionné par une fonction ou d'autres variables), ça n'est pas bloquant.



Dans de nombreux cas, la fonction `is_defined` peut avantageusement être remplacée par la fonction `getVar` à laquelle on aura pris soin d'indiquer une valeur par défaut à renvoyer en cas d'indisponibilité de la variable (voir ci-dessous).

Variable "vide" : `is_empty`

Il n'est pas toujours évident, en particulier lorsque l'on manipule des variables multi-valuées, de trouver le test adéquat afin de déterminer si une variable est vide.

Pour tester si une variable est vide, il est désormais recommandé d'utiliser la fonction `%%is_empty`.



```
%%if not %%is_empty(%%ma_variable)
%%ma_variable[0]
%%else
la variable est vide
%%end if
```

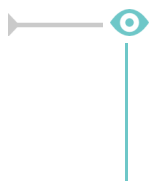
Concaténation des éléments d'une liste : `custom_join`

La fonction `%%custom_join` permet de concaténer facilement les éléments d'une variable multi-valuée.

Cela permet d'éviter le recours à une boucle `%%for` et l'utilisation de l'instruction `%%slurp` qui est souvent source d'erreurs.

Il est possible de spécifier le séparateur à utiliser en le passant comme paramètre à la fonction.

En l'absence de ce paramètre, le séparateur utilisé est l'espace.



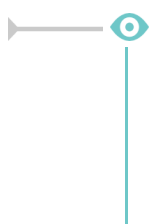
```
%%custom_join(%%ma_variable, ':')
```

Si `ma_variable` vaut ['a', 'b', 'c'], cela donnera :

```
a:b:c
```

Variable "dynamique" : `getVar`

Une variable dynamique prend comme nom (ou partie du nom) la valeur d'une autre variable.



```
%%for %%interface in range(0, %%int(%%nombre_interfaces))
L'interface eth%%interface a pour adresse
%%getVar('adresse_ip_eth'+str(%%interface))
%%end for
```

La fonction *getVar* peut également être utilisée lorsque l'on n'est pas certain qu'une variable est disponible (inexistante ou désactivée) car il est possible de lui spécifier une valeur par défaut à renvoyer en cas d'indisponibilité.

```
%if %%getVar("activer mon logiciel", "non") == 'oui'
  Activation du logiciel
%end if
```

Variable esclave "dynamique" : getattr

Lorsque le nom de la variable esclave doit être calculé, on peut utiliser `%%getattr` à la place de la notation pointée.

```
%set %%num='0'
%for %%ip_ssh in %%getVar('ip_ssh_eth'+%%num)
  SSH est autorisé pour %%ip_ssh/%%getattr(%%ip_ssh,
  'netmask_ssh_eth'+%%num)
%end for
```

La fonction *getattr* peut également être utilisée lorsque l'on n'est pas certain qu'une variable esclave est disponible (inexistante ou désactivée) car il est possible de lui spécifier une valeur par défaut à renvoyer en cas d'indisponibilité.

```
%for %%iterator %%var master
  %%getattr(%%iterator, 'var_slave', 'default')
%end for
```

Autres fonctions

Fonctions de traitement des chaînes de caractères

- transformation d'une chaîne en majuscules : `%%upper(%%ma_chaine)` ;
- transformation d'une chaîne en minuscules : `%%lower(%%ma_chaine)` ;
- encodage d'une chaîne en ISO-8859-1 (au lieu d'UTF-8) : `%%to_iso(%%ma_chaine)` ;
- transformation d'un masque réseau (ex : 255.255.255.0) en classe d'adresse (ex : 24) : `%%calc_classe(%%mask)` ;

Fonctions de tests

- vérification que la variable est une adresse IP (et pas un nom DNS) : `%%is_ip(%%variable)` ;
- vérification de l'existence d'un fichier : `%%is_file(%%fichier)`.

Déclaration de fonctions locales

Pour un traitement local et répétitif, il peut être pratique de déclarer une fonction directement dans un template avec `%def` et `%end def`.

Cependant, la syntaxe à utiliser dans ces fonctions est assez complexe (on ne sait jamais quand mettre le caractère `%` !) et ce genre de déclaration ne facilite pas la lisibilité du template.

Les fonctions déclarées localement s'utilisent de la même façon que les fonctions déjà prédéfinies.



```
%def nombre_points(chaine)
..%return chaine.count('.')
%end def

Il y a %%nombre_points(%%ma variable) points dans ma variable.
```

Ajout de fonctions personnalisées

Il est possible d'ajouter des librairies de fonctions personnalisées dans le répertoire `/usr/share/creole/funcs`.

Les librairies doivent posséder l'extension `.py` et contenir des fonctions python.



```
# -*- coding: utf-8 -*-
def to_iso(data):
    """ encode une chaine en ISO """
    try:
        return unicode(data, "UTF-8").encode("ISO-8859-1")
    except:
        return data
```



Si vous devez importer des librairies python dans un fichier de fonctions personnalisées, ne les importez pas en début de fichier. Les imports doivent être faits dans la fonction de calcul elle-même.



Les adaptations que vous pouvez réaliser sur l'un de vos serveurs EOLE sont susceptibles d'intéresser d'autres utilisateurs. Elles peuvent faire l'objet d'une intégration dans le projet EOLE par l'équipe de développement.

Les avantages sont multiples :

- pérennité de vos modifications ;
- diffusion sur l'ensemble de vos serveurs ;
- optimisé par l'équipe ;
- diffuser à tous les utilisateurs.

Aussi n'hésitez pas à proposer votre travail. Pour se faire vous pouvez vous référer à la documentation pour apprendre comment contribuer.

3. Utilisation avancée

Modification des méta-caractères utilisés

Dans le cas où il y a trop de % dans le template, il est possible de changer carrément de méta-caractères, en ajoutant une section `compiler-settings` en en-tête du template.

Cette méthode est, par exemple, utilisée pour la génération du fichier de configuration du logiciel `eJabberd` qui est en déclaré en Erlang^[p.64].



Utilisation de @ et @@ à la place de % et %%

```
%compiler-settings
directiveStartToken = @
cheetahVarStartToken = @@
%end_compiler-settings
```

Utilisation de `creole_client`

Les fonctionnalités de `creole_client` sont utilisables directement dans les templates.

Il est par exemple possible de lister toutes les variables et leurs valeurs :

```
%for %%var, %%value in %%creole client.get creole().items()
. %%var : %%value
%end for
```

Donnera le résultat suivant (notez que le nom des variables esclaves est précédé de celui de la variable maître associée) :

```
ssl_organization_name : Ministere Education Nationale (MENESR)
https port :
check passwd min len two type : 9
container ip proxy : 127.0.0.1
nom cache pere zone.options cache pere zone : []
nom cache pere : []
ignore_expect_100 :
off eolessa adresse : 192.168.230.205
activer dhcprelay : non
[ ... ]
```

Plus généralement, il est possible d'accéder à toutes les informations décrites dans les dictionnaires comme celles concernant les conteneurs, les services et les tâches programmées.

Liste des conteneurs :

```
%for %%container in %%creole client.get containers()
* %%container['name']
%end for
```

Liste des services actifs :

```
%for %%srv in %%creole client.get services()
%if %%srv.has key('activate')
* %%srv['name']
%end if
%end for
```

```
%set %%sched = %%creole client.get('schedule.schedule')
```

```
Les tâches programmées sont exécutées à
%%{sched['hour']}h%%{sched['minute']}
```

4. Exemple

▶ Templatiser un nouveau fichier

Nous voulons templatiser le fichier `toto.conf` à l'aide des mécanismes Creole afin de rajouter l'`adresse_ip_eth0` (variable existante) ainsi que l'adresse de l'établissement (nouvelle variable).

● Ajouter un dictionnaire local

Dans `/usr/share/eole/creole/dicos/local/`

ajouter un fichier `.xml`

● Ajouter votre fichier template

Notre fichier `toto.conf` sera placé dans `/usr/share/eole/creole/distrib/`

Il faut ajouter les variables à l'aide de la syntaxe Creole.

exemple : l'adresse est `%%adresse_ip_eth0` et l'adresse est `%%adresse_etablissement`

● Entrer l'adresse de l'établissement

- Aller dans l'interface de configuration du module
- Dans l'onglet `Perso` renseigner l'adresse de l'établissement
- Enregistrer

● Reconfigurer

Le mécanisme de configuration a écrit votre fichier `/etc/toto.conf` avec les variables.

💬 Commentaires généraux

Les variantes Zéphir

Cette procédure décrit comment ajouter des spécifications locales.

Dans le cadre d'un développement massif, le module Zéphir propose un mécanisme de variantes semblable.

Instancier un template avec CreoleCat

CreoleLint et CreoleCat ^[p.41]

Chapitre 6

Les scripts Creole

Creole fournit également un ensemble de scripts destinés à faciliter l'administration du serveur :

- `CreoleLint` permettant de faire des vérifications sur un dico ou sur un template ;
- `CreoleCat` permettant d'instancier un seul template indépendamment des commandes `instance` et `reconfigure` ;
- `CreoleGet` et `CreoleSet` permettant de lire et de modifier la valeur d'une variable Creole.
- `CreoleRun` et `CreoleService` permettant de lancer des commandes système et de gérer les services sur les modules EOLE, y compris à l'intérieur des conteneurs^[p.63] ;
- `CreoleLock` permettant de placer, enlever ou vérifier les verrous Creole.

1. CreoleLint et CreoleCat

`CreoleLint` et `CreoleCat` sont des utilitaires permettant de faciliter les tests sur les dictionnaires et les templates :

- `CreoleLint` permet de valider la syntaxe des dictionnaires et des templates ;
- `CreoleCat` permet d'instancier un seul template indépendamment des commandes `instance` et `reconfigure` .

Vérifier les dictionnaires et templates avec CreoleLint

La commande `CreoleLint` permet de valider la syntaxe des dictionnaires et des templates.

L'outil effectue une série de tests dans le but de détecter des erreurs dans la déclaration et l'utilisation des variables.

Sur un module installé, il est possible de lancer l'application sans option particulière :

```
# CreoleLint
```

Cette commande permet également :

- de valider un seul template avec l'option `-t` : `CreoleLint -t hostname`
- de ne lancer qu'un seul des tests lint avec l'option `-n nomDuTest` : `CreoleLint -n valid dtd`
- de ne lancer que la validation des dictionnaires avec l'option `-d` : `CreoleLint -d`

Les tests lint disponibles sont les suivants :

- `valid dtd` : validation syntaxique des dictionnaires ;
- `tabs in dicos` : recherche de tabulation dans les dictionnaires ;
- `hidden if in dicos` : recherche des conditions dépréciées `hidden if in` et

```
hidden if not in ;
```

- `obligatoire in dicos` : recherche du validateur déprécié `obligatoire` ;
- `valid slave value` : recherche les variables esclaves avec une liste en valeur défaut (EOLE >= 2.5.2) ;
- `wrong dicos name` : validation du nom des dictionnaires ;
- `valid var label` : vérification des libellés des variables ;
- `valid separator label` : vérification des libellés des séparateurs ;
- `valid help label` : vérification des libellés de l'aide en ligne ;
- `activation var without help` : vérification des variables d'activation sans balise d'aide (EOLE >= 2.5.2) ;
- `family without help` : vérification des familles sans balise d'aide ;
- `family without icon` : vérification des familles sans icône spécifique ;
- `old fw file` : recherche des anciens fichiers eole-firewall ;
- `valid parse tpl` : validation de tous les templates.



L'option `-l` permet de choisir le niveau des messages (info, warning ou error).

La commande `CreoleLint` suivie du paramètre `-h` permet d'obtenir de l'aide. Un manuel est également disponible :

```
# man CreoleLint
```

Instancier un template avec CreoleCat

La commande `CreoleCat` permet d'instancier un seul template indépendamment des commandes `instance` et `reconfigure`.

Cette commande permet :

- d'instancier un seule template existant sur le module en utilisant la ou les destinations déclarées dans le dictionnaire :

```
# CreoleCat -t hostname
```

- d'instancier un template existant sur le module en redirigeant le résultat dans un fichier spécifique :

```
# CreoleCat -t hostname -o /tmp/hostname.txt
```

- d'instancier un fichier template spécifique en redirigeant le résultat dans un fichier spécifique :

```
# CreoleCat -s /tmp/test.tpl -o /tmp/test.txt
```

- d'instancier un fichier template spécifique en affichant le résultat sur la console (EOLE >= 2.5.2) :

```
# CreoleCat -s /tmp/test.tpl
```



L'option `-l` permet de choisir le niveau des messages (info, warning ou error).

Les options `-v` (`--verbose`) ou `-d` (`--debug`) permettent de connaître le détail des

opérations réalisées par le programme.

La commande `CreoleCat` suivie du paramètre `-h` permet d'obtenir de l'aide.

```
root@scribe:~# CreoleCat -d -t sympa.auth.conf
Instanciation du fichier '/etc/sympa/auth.conf' depuis
'/var/lib/creole/sympa.auth.conf'
Copy template: '/usr/share/eole/creole/distrib/sympa.auth.conf' ->
'/var/lib/creole'
Cheetah processing: '/var/lib/creole/sympa.auth.conf' ->
'/etc/sympa/auth.conf'
Changing properties: chown sympa:sympa /etc/sympa/auth.conf
Changing properties: chmod 0644 /etc/sympa/auth.conf
```

Dans le cas d'un template renommé, c'est le nom du template (défini dans l'attribut *source*) qu'il faut utiliser.

2. CreoleGet et CreoleSet

CreoleGet et **CreoleSet** sont des utilitaires permettant de lire et de modifier la valeur d'une variable Creole.

Récupérer la valeur d'une variable avec CreoleGet

CreoleGet est un utilitaire très pratique pour récupérer la valeur d'une variable Creole.

Il s'utilise tout simplement en lui donnant le nom de la variable souhaitée en argument :

```
CreoleGet mvariable
```

La commande `CreoleGet --list` permet d'obtenir la liste complète des variables.

La commande `CreoleGet` supporte l'autocomplétion à partir de la version 2.5.2 d'EOLE.

```
# CreoleGet --list | grep release
eole_release="2.4.2"
```

CreoleGet permet également de récupérer la liste des groupes de conteneurs :

```
CreoleGet --groups
```

Sur un serveur en mode non conteneur, cette commande renvoie uniquement `root`.



Dans le cas où l'on n'est pas certain que la variable soit disponible (variable inconnue ou désactivée), il est possible d'indiquer une valeur par défaut à renvoyer en cas d'erreur :

```
CreoleGet activer_logiciel non
```

Dans le cas contraire, une erreur pourra apparaître.



Pour accéder à une variable esclave, il faut connaître la variable maître :

```
CreoleGet lamaster.lesclave
```



Les valeurs multiples sont séparées par un saut de ligne (`\n`) :

```
root@eolebase:~# CreoleGet serveur maj
eole.ac-dijon.fr
ftp.crihan.fr
```



L'option `-h` ou `--help` ou la commande `man CreoleGet` permettent d'obtenir de l'aide.

Lister les services gérés par Creole avec CreoleGet

La commande suivante permet d'obtenir la liste des services qui sont gérés par CreoleService sur le module :

```
CreoleGet .containers.services |grep \.name=
```



```
1 root@eolebase:~# CreoleGet .containers.services|grep \.name=
2 service0.name="networking"
3 service1.name="cron"
4 service10.name="exim4"
5 service11.name="eoleflask"
6 service12.name="nginx"
7 service13.name="ead3"
8 service14.name="genconfig"
9 service15.name="bastion"
10 service16.name="z_stats"
11 service2.name="rng-tools"
12 service3.name="ntp"
13 service4.name="nut-server"
14 service5.name="salt-api"
15 service6.name="salt-master"
16 service7.name="salt-minion"
17 service8.name="ead-server"
18 service9.name="ead-web"
19 root@eolebase:~#
```

Modifier la valeur d'une variable avec CreoleSet

CreoleSet est un utilitaire très pratique pour modifier la valeur d'une variable Creole.

Il s'utilise tout simplement en lui donnant le nom de la variable et sa valeur en argument :

```
CreoleSet mon_ip 10.10.10.55
```



L'option `--default` permet de réinitialiser une variable à sa valeur par défaut :

```
CreoleSet --default serveur_ntp
```

La commande `CreoleSet` supporte l'autocomplétion à partir de la version 2.5.2 d'EOLE.



Les valeurs multiples doivent être séparées par un saut de ligne (`\n`) :

```
root@eolebase:~# CreoleSet serveur_maj "eole.ac-toto.fr
ftp.crihan.fr"
```



La modification d'une variable possédant des dépendances fortes avec d'autres variables ou familles ne sera généralement pas possible car cela cassera la consistance des données.



L'option `-h` ou `--help` ou la commande `man CreoleSet` permettent d'obtenir de l'aide.

3. CreoleRun et CreoleService

CreoleRun et **CreoleService** sont des utilitaires permettant de lancer des commandes système et de gérer les services sur les modules EOLE, y compris à l'intérieur des conteneurs^[p.63].

Exécuter une commande avec CreoleRun

CreoleRun est un utilitaire très pratique pour exécuter une commande dans un conteneur (depuis le maître).

Le script s'utilise de la façon suivante :

```
CreoleRun "<command>" <container>
```



Si le mot clé `all` est utilisé à la place du nom du conteneur, alors la commande sera lancée dans tous les conteneurs (rien ne sera exécuté en mode non conteneur).

La commande gère un troisième argument qui si il vaut `yes` exécutera la commande uniquement si l'environnement est un conteneur (ie : si l'utilisation de SSH est nécessaire).

Gérer les services avec CreoleService

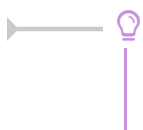
CreoleService permet de gérer les services déclarés dans les dictionnaires Creole.

Le script s'utilise de la façon suivante :

```
CreoleService [-c <container>] <service> <action>
```

Les actions possible sont :

- *configure* : configure le lancement automatique du service au démarrage du serveur en fonction de la configuration Creole du serveur ;
- *enable* : active le lancement automatique du service au démarrage du serveur ;
- *disable* : désactive le lancement automatique du service au démarrage du serveur ;
- *apply* : démarre ou arrête le service en fonction de la configuration Creole du serveur ;
- *start* : démarre le service ;
- *stop* : arrête le service ;
- *restart* : redémarre le service ;
- *reload* : recharge le service ;
- *status* : vérifie l'état du service.



L'option, `-f` (ou `--force`) permet de forcer le démarrage ou redémarrage d'un service même si celui-ci est désactivé au niveau de la configuration Creole du serveur.

4. CreoleLock

CreoleLock est un utilitaire permettant de placer, enlever ou vérifier les verrous Creole.

Il peut gérer deux niveaux (level) de verrouillage distincts.

La plupart des outils de base EOLE utilisent de verrous de niveau "système".

Verrou "normal"

Ce type de verrou permet d'éviter qu'une même application soit exécutée deux fois en parallèle. Il s'agit donc d'un verrou isolé.

En mode normal (`--level=normal`), les fichiers lock sont écrits dans le répertoire `/var/lock/eole` et il est possible d'exécuter plusieurs applications différentes en même temps tant qu'elles ne posent pas un lock ayant le même nom.

Verrou "système"

Contrairement au mode normal, les verrous "système" (`--level=system`) sont exclusifs. Cela permet d'éviter que deux applications concurrentes sont exécutées en même temps. Par exemple, il ne faut pas qu'un reconfigure soit exécuté en même temps qu'une sauvegarde : ces deux procédures utilisent des verrous "système".

Dans ce mode, mes fichiers lock sont écrits dans le sous-répertoire `/var/lock/eole/eole-system`.

Nom d'un fichier lock

Le nom d'un fichier lock est de la forme `prefixe.suffixe`, avec :

- un préfixe invariant fourni par le programme (généralement le nom de l'application) ;
- un suffixe représentant le PID^[p.64] de l'application.

Poser un verrou avec CreoleLock

Pour poser un verrou nommé *toto*, la commande à taper est la suivante :

```
CreoleLock acquire --name toto
```

Si un verrou existe déjà, la commande affichera un message d'erreur et ne renverra pas le code `0`.

Vérifier la présence d'un verrou avec CreoleLock

Pour vérifier la présence du verrou nommé *toto*, la commande à taper est la suivante :

```
CreoleLock is_locked --name toto
```

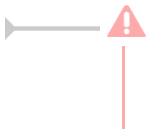
Cette commande retournera le code `0` si le verrou est présent.

Supprimer un verrou avec CreoleLock

Pour supprimer un verrou nommé *toto*, la commande à taper est la suivante :

```
CreoleLock release --name toto
```

Cette commande retournera le code `0` en cas de succès.



Seul le programme (y compris la console si la commande est lancée en console) qui a posé le verrou a le droit de le supprimer.

API python

La librairie `pyeole.lock` permet de gérer les verrous Creole directement en python.

Elle fournit notamment les fonctions `acquire`, `is_locked` et `release`.



L'option `-h` permet d'afficher les paramètres de la commande CreoleLock :

```
# CreoleLock -h
usage: /usr/bin/CreoleLock [acquire|release|is_locked]
[options|--help]
```

5. Indications pour la programmation

Certaines fonctions ont été intégrées sur les modules afin que les scripts puissent être écrits en tenant compte des spécificités des modules EOLE, que sont les variables et le mode conteneur.

Programmation bash

- obtenir la valeur d'une variable (variables de conteneur comprises) :

```
CreoleGet <variable_name>
```

- obtenir la valeur d'une variable ou une valeur prédéfinie en cas d'erreur :

`CreoleGet <variable_name> <default_value>`

- modifier la valeur d'une variable :

`CreoleSet <variable_name> <new_value>`

- exécution d'une commande dans un conteneur :

`CreoleRun "<command>" <container>`

- redémarrage d'un service dans un conteneur :

`CreoleService -c <container> <service name> restart`

Petit script bash

```
1#!/bin/bash
2echo "mon adresse IP est $(CreoleGet adresse_ip_eth0)"
3echo "La base Ldap est stockée dans $(CreoleGet container_path_annuaire)
  /var/lib/ldap"
4echo "Le conteneur annuaire a l'adresse : $(CreoleGet
  container_ip_annuaire)"
5CreoleRun "ls /var/lib/ldap" annuaire
6CreoleService slapd restart -c annuaire
```

Script compatible EOLE 2.3/2.4/2.5

```
1#!/bin/bash
2if [ -f /usr/bin/ParseDico ];then
3  RunCmd=RunCmd
4  . /usr/bin/ParseDico
5  . /etc/eole/containers.conf
6  . /usr/share/eole/FonctionsEoleNg
7else
8  RunCmd=CreoleRun
9  # récupération des variables nécessaires
10 container_path_web=$(CreoleGet container_path_web)
11 nom_machine=$(CreoleGet nom_machine)
12fi
13touch ${container_path_web}/etc/${nom_machine}.conf
14$RunCmd "chown www-data /etc/${nom_machine}.conf" web
```



`CreoleGet` permet également d'accéder aux variables "extra" :

`CreoleGet schedule.schedule.hour`

Programmation Python

- obtenir la valeur d'une variable (variables de conteneur comprises) :

```
from creole.client import CreoleClient
CreoleClient().get creole('<variable name>')
```

- obtenir la valeur d'une variable ou une valeur prédéfinie en cas d'erreur :

```
from creole.client import CreoleClient
CreoleClient().get creole('<variable name>', '<default value>')
```

- obtenir l'ensemble des variables dans un dictionnaire :

```
from creole.client import CreoleClient
dico = CreoleClient().get creole()
```



```
adresse_ip_eth0 = dico['adresse_ip_eth0']
```

cas particulier: pour les variables 'esclaves' d'un groupe, préfixer par la variable maître

```
sso_first_base_ldap = dico['eolessso_ldap.eolessso_base_ldap'][0]
```

- obtenir la valeur d'une esclave correspond à une master :

```
master = client.get_creole('master')
```

```
slave = client.get_creole('slave')
```

```
for idx, var in enumerate(master):
```

```
print "master : {0}, slave : {1}".format(var, slave[idx])
```

- exécution d'une commande dans un conteneur (affichage à l'écran) :

```
from pyeole.process import system_code
```

```
system_code([<commande sous forme de liste>], container='<conteneur>')
```

- exécution d'une commande dans un conteneur (sorties dans un tuple) :

```
from pyeole.process import system_out
```

```
system_out([<commande sous forme de liste>], container='<conteneur>')
```

- redémarrage d'un service dans un conteneur (avec affichage à l'écran)

```
from pyeole.log import init_logging
```

```
from pyeole.service import manage_service
```

```
init_logging(level='info')
```

```
manage_service('restart', '<service>', '<conteneur>')
```

Petit script Python

```
1#!/usr/bin/env python
2# -*- coding: UTF-8 -*-
3from creole.client import CreoleClient
4creole_client = CreoleClient()
5print "mon adresse IP est {0}".format(creole_client.get_creole(
6    'adresse_ip_eth0'))
7print "La base Ldap est stockée dans {0}/var/lib/ldap".format(
8    creole_client.get_creole('container_path_annuaire'))
9print "Le conteneur annuaire a l'adresse : {0}".format(creole_client.
10    get_creole('container_ip_annuaire'))
11from pyeole.process import system_code
12system_code(['ls', '/var/lib/ldap'], container='annuaire')
13from pyeole.log import init_logging
14from pyeole.service import manage_service
15init_logging(level='info')
16manage_service('restart', 'slapd', 'annuaire')
```

Script compatible EOLE 2.3/2.4/2.5

```
1#!/usr/bin/env python
2# -*- coding: UTF-8 -*-
3from pyeole.process import system_code
4try:
5    from creole import parsedico
6    from creole.eosfunc import load_container_var
7    variables = parsedico.parse_dico()
8    variables.update(load_container_var())
9except:
```

```

10     from creole.client import CreoleClient
11     variables = CreoleClient().get_creole()
12 fichier = open('{0}/etc/{1}.conf'.format(variables['container_path_web'],
    variables['nom_machine']), 'a')
13 fichier.close()
14 system_code(['chown', 'www-data', '/etc/{0}.conf'.format(variables[
    'nom_machine'])], container='web')

```

Modification de variables

Du fait des dépendances entre variables certaines modifications ne sont pas réalisables avec la commande `CreoleSet`.

C'est notamment le cas pour les variables groupées qui doivent impérativement posséder le même nombre d'éléments au moment de l'enregistrement ou pour des variables de type `oui/non` qui permettent de débloquent des variables à caractère obligatoire.

L'exemple qui suit montre comment activer l'autorisation des connexion SSH pour un couple adresse IP / masque de sous-réseau.

```

1 #!/usr/bin/env python
2 # -*- coding: UTF-8 -*-
3 from creole.loader import creole_loader, config_save_values
4 config = creole_loader(rw=True)
5 config.creole.interface_0.ssh_eth0 = u'oui'
6 config.creole.interface_0.ip_ssh_eth0.ip_ssh_eth0[0] = u'192.168.1.1'
7 config.creole.interface_0.ip_ssh_eth0.netmask_ssh_eth0[0] =
    u'255.255.255.255'
8 config_save_values(config, 'creole')

```

Pour accéder à une variable esclave, il faut connaître le nom de sa famille et celui de la variable maître associée.

Les valeurs doivent être saisies en Unicode^[p.65], qui en python se traduit par l'ajout du caractère `u` devant la chaîne de caractères.

Cette obligation ne concerne pas les variables de type `number` qui attendent un nombre entier :

```
config.creole.systeme.bash_tmout = 3600
```

Chapitre 7

Ajout de script exécuté à l'instance ou au reconfigure

Il est parfois nécessaire d'ajouter un script qui sera exécuté à l'instanciation ou au reconfigure du module. EOLE met en place des mécanismes permettant d'exécuter des scripts avant ou après l'instanciation ou la reconfiguration.

Ces scripts doivent être dans l'un des répertoires suivants :

- `/usr/share/eole/preservice` : exécution avant l'arrêt des services ;
- `/usr/share/eole/pretemplate` : exécution avant la templatisation des fichiers ;
- `/usr/share/eole/posttemplate` : exécution entre la templatisation des fichiers et le redémarrage des services ;
- `/usr/share/eole/postservice` : exécution après le redémarrage des services.



Chacun des scripts doit respecter les contraintes exigées par l'outil `run-parts`, et, en particulier :

- être exécutable ;
- être sans extension.



L'ensemble de ces scripts se jouent de façon alphanumérique.

Les scripts fournis par EOLE sont préfixés par des chiffres et un tiret :

```
1 root@scribe:/usr/share/eole/preservice# ll
2 total 28
3 drwxr-xr-x  2 root root 4096 sept. 28 10:24 ./
4 drwxr-xr-x 29 root root 4096 sept. 28 10:24 ../
5 -rwxr-xr-x  1 root root  387 sept. 28 09:16 00-anetwork*
6 -rwxr-xr-x  1 root root  464 sept.  7 15:08 00-bareoswebui*
7 -rwxr-xr-x  1 root root  500 juin 26 2015 00-save-sid*
8 -rwxr-xr-x  1 root root  702 sept.  7 15:36 00-web*
9 -rwxr-xr-x  1 root root  235 sept. 28 09:16 99-ifupdown*
10 root@scribe:/usr/share/eole/preservice#
```

Le type d'appel (instance ou reconfigure) est envoyé au script sous la forme d'un argument :

```
1 #!/bin/bash
2 if [ "$1" == "instance" ]; then
3     echo "ce code n'est exécuté qu'à l'instance"
4 elif [ "$1" = "reconfigure" ] ;then
5     echo "ce code n'est exécuté qu'au reconfigure"
6 fi
```



Si le script quitte avec un autre code de retour que `0`, l'instance ou le reconfigure s'arrête immédiatement.

Il est donc préférable que le script soit de la forme :

```
1 #!/bin/bash
2 # <<< SCRIPT >>>
3 exit 0
```

Voir aussi...

Indications pour la programmation [p.47]

Chapitre 8

Ajout d'un test diagnose

Les scripts diagnose personnalisés peuvent être placés dans le répertoire `/usr/share/eole/diagnose`. Ces fichiers sont généralement écrits en bash et permettent de se connecter au service voulu pour tester l'état de celui-ci.



Chacun des scripts doit respecter les contraintes exigées par l'outil `run-parts`, et, en particulier :

- être exécutable ;
- être sans extension.

Un certain nombre de fonctions sont disponibles dans les bibliothèques EOLE, mais vous pouvez créer vos propres fonctions pour vos besoins spécifiques.

Généralement, le test affiche *Ok* si le service est fonctionnel et *Erreur* en cas de problème.

Voici quelques fonctions disponibles dans la bibliothèque `/usr/lib/eole/diagnose.sh` :

- *TestIP* et *TestIP2* : testent si une IP répond au ping ;
- *TestARP* : teste si l'adresse MAC associée à une IP répond ;
- *TestService* : teste la connexion TCP sur une IP et un numéro de port ;
- *TestUDP* : teste si un port est ouvert localement en UDP ;
- *TestPid* : teste la présence du PID d'une application locale ;
- *TestDns* : teste la résolution de nom sur un serveur DNS particulier ;
- *TestNTP* : teste un serveur NTP ;
- *TestHTTPPage* : teste l'ouverture d'une session HTTP ;
- *TestWeb* : teste le téléchargement d'une page HTTP ;
- *TestCerts* : teste des valeurs du certificat TLS/SSL.



```
#!/bin/bash
# utilisation des fonctions EOLE
. /usr/lib/eole/diagnose.sh
# teste si le serveur web local est fonctionnel
# en vérifiant la variable Creole "activer apache"
# et en utilisant la fonction TestHTTPPage
if [ $(CreoleGet activer apache) = "oui" ];then
    TestHTTPPage "Web local" "http://$(CreoleGet
adresse_ip_eth0)/"
```

| fi

Voir aussi...

Indications pour la programmation [p.47]

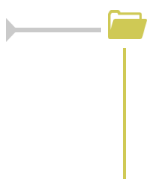
Chapitre 9

Gestion des noyaux Linux

Noyau Linux utilisé

Les modules EOLE 2.5 utilisent par défaut le noyau le plus récent de la distribution Ubuntu.

Si le noyau utilisé est différent du noyau conseillé, les commandes `instance` et `reconfigure` vous proposeront de redémarrer le serveur ou le redémarreront automatiquement en fonction de la situation.



Sur les dernières versions d'Ubuntu 14.04, le noyau utilisé est `linux-image-generic-lts-vivid`.

Pour plus d'informations, consulter la page : <http://doc.ubuntu-fr.org/ltsenablementstack>

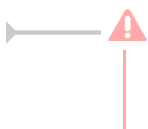


La commande `uname -r` permet de connaître le noyau en cours d'utilisation.

En-tête du noyau

Plusieurs outils nécessitent la présence des en-têtes du noyau (headers) sur le serveur.

Les en-têtes du noyau courant sont pré-installés sur les modules.



Les en-têtes des anciens noyaux sont purgés automatiquement lorsque le noyau associé est supprimé.

Purge des anciens noyaux

Tous les noyaux sont purgés à l' `instance` et au `reconfigure` à l'exception :

- du noyau en cours d'utilisation ;
- du noyau précédent le noyau utilisé ;
- du noyau le plus récent installé ;
- d'un éventuel noyau personnalisé (voir ci-dessous).

Personnalisation du noyau

Dans certains cas (prise en charge de matériels, tests,...), il peut être nécessaire d'utiliser un autre noyau (compilé ou non par vos soins) que le noyau courant.

Créer le fichier `/usr/share/eole/noyau/local` avec le numéro de version du noyau à utiliser permet de forcer l'utilisation d'un noyau antérieur ou d'un noyau compilé.



Pour utiliser le noyau **linux-image-4.4.0-112-generic** il faut ajouter le numéro de version du noyau 4.4.0-112 dans le fichier `/usr/share/eole/noyau/local` :

```
# echo 4.4.0-112 > /usr/share/eole/noyau/local
```

Mettre à jour Grub :

```
# update-grub
```

Pour réutiliser le noyau courant il faut supprimer le fichier `/usr/share/eole/noyau/local` et mettre à jour Grub à l'aide de la commande `update-grub` .



Cette facilité est à utiliser à titre exceptionnel.

Aucun signalement lié à l'utilisation d'un noyau différent de celui préconisé par EOLE ne sera pris en compte.

Chapitre 10

Gestion des tâches planifiées eole-schedule

Présentation

Sur les modules EOLE, les tâches planifiées (comme par exemple les mises à jour, les sauvegardes, la purge de certaines informations, l'exportation de l'annuaire, des bases de données et des quotas disque ou encore les mises à des listes noires pour le filtrage proxy) sont gérées par `eole-schedule`.

Contrairement à l'utilisation de cron, `eole-schedule` permet de maîtriser les tâches planifiées même si la sauvegarde est activée.

En version 2.5, `eole-schedule` est géré depuis Tiramisu^[p.64].

Le principe est le suivant :

- si aucune sauvegarde n'est prévue, c'est cron^[p.63] qui lance `eole-schedule` ;
- si une sauvegarde est prévue, c'est Bareos^[p.63] qui lance `eole-schedule`.

Il existe 4 types de tâches planifiées :

- les tâches journalières : *daily* ;
- les tâches hebdomadaires : *weekly* ;
- les tâches mensuelles : *monthly* ;
- les tâches uniques : *once*.

Ces tâches sont découpées en *pre*-sauvegarde et *post*-sauvegarde.

Si aucune sauvegarde n'est prévue : le *cron* lance *pre* puis *post* à l'heure qui a été tirée au hasard.

Si une sauvegarde est prévue : Bareos lance *pre* avant la sauvegarde et *post* à l'heure qui a été tirée au hasard (sauf si celle-ci est prévue avant la sauvegarde ou si la sauvegarde n'est pas terminée, dans ce cas les tâches *post* sont exécutées après la sauvegarde).

Les sauvegardes "post" sont obligatoirement marquées en `Full` même si cela ne correspond à rien (pas de sauvegarde, exécution des scripts uniquement). Elles sont réalisées à l'heure qui a été tirée au hasard.

Par contre, les sauvegardes "pre" sont bien lancées à l'heure des sauvegardes définie par l'administrateur.

Différences par rapport à Schedule 2.3

La liste des scripts à activer est décrite dans un fichier XML^[p.65] (dictionnaire). Ce système permet de mettre en place des valeurs par défaut. Ainsi, l'activation ou la désactivation d'un script n'est plus réalisée à l'installation du paquet ce qui est à la fois plus simple et plus sûr.

La description n'est plus dans le script. Elle est directement dans le fichier XML.

Les scripts *pre/post* sont maintenant mélangés dans le répertoire `/usr/share/eole/schedule/scripts`.

Gestion des tâches planifiées

Lister ce qui est programmé

```
# manage_schedule -l
```

Ajouter une tâche planifiée

```
# manage_schedule -a daily -s majblacklist -m post
```

Supprimer une tâche planifiée

```
# manage_schedule -d majblacklist
```

Appliquer la configuration (génération des liens symboliques)

```
# manage_schedule --apply
```



L'ajout et la suppression n'appliquent pas la configuration. Il faut :

- soit l'appliquer à la main (`manage_schedule --apply`) ;
- soit effectuer un `reconfigure` .

Gestion des tâches uniques (once)

Les scripts lancés pour une nuit sont gérés totalement différemment et les informations associées ne sont pas conservées dans Tiramisu.

Ajouter une tâche planifiée unique

```
# manage_schedule -a once -s majauto -m post
```

Supprimer une tâche planifiée unique

```
# manage_schedule -d once -s majauto -m post
```



La prise en compte des tâches uniques est instantanée.

L'appel à la méthode `--apply` n'est donc pas nécessaire.

Exemple de fichier XML

Les fichiers XML décrivant les tâches planifiées ont un format proche de celui des dictionnaires^[p.63] Creole.



Exemple du fichier : `/usr/share/eole/creole/extra/schedule/01_majauto.xml`

```
1 <?xml version="1.0" encoding="utf-8"?>
2
3 <creole>
4   <variables>
5     <family name='majauto'>
```

```

6         <variable name="description" type="string"><value>Mise à jour
          du serveur</value></variable>
7         <variable name="day" type="schedule"><value>weekly
          </value></variable>
8         <variable name="mode" type="schedulemod"><value>post
          </value></variable>
9         </family>
10        </variables>
11 </creole>

```

Gestion des mises à jour avec Creole et eole-schedule

La mise à jour hebdomadaire consiste en un script `eole-schedule` nommé `majauto`. Il est configuré pour être lancé une fois par semaine (`weekly`) après la sauvegarde (`post`).

Sa gestion dans les scripts python est facilitée par la librairie `creole.maj`.

⚡ Savoir quand est prévue la mise à jour

```
# python -c "from creole import maj; print maj.get_maj_day()"
```

⚡ Activer/désactiver la mise à jour hebdomadaire

Activation de la mise à jour hebdomadaire :

```
# manage_schedule -a weekly -s majauto -m post
```

ou :

```
# python -c "from creole import maj; maj.enable_maj_auto(); print maj.maj_enabled()"
```

Désactivation de la mise à jour hebdomadaire :

```
# manage_schedule -d majauto
```

ou :

```
# python -c "from creole import maj; maj.disable_maj_auto(); print maj.maj_enabled()"
```

Forcer l'exécution des tâches planifiées

Il est possible de forcer l'exécution des tâches planifiées avec la commande `/usr/share/eole/schedule/schedule cron`.

```

1 root@amon:~# /usr/share/eole/schedule/schedule cron
2 Démarrage de pre schedule daily
3 pre schedule daily accompli
4 Démarrage de post schedule daily
5 . Test de http://eole.orion.education.fr/maj/blacklists => Ok
6 Téléchargement des bases
7 Rien à faire pour blacklists.tar.gz
8 Rien à faire pour le fichier weighted
9 eole-schedule - run-parts: executing
  /usr/share/eole/schedule/daily/post/majblacklist daily
10 post schedule daily accompli
11 Démarrage de pre schedule once
12 pre schedule once accompli
13 Démarrage de post schedule once
14 post schedule once accompli
15 root@amon:~#

```

Lire les journaux de l'exécution des tâches planifiées

Les journaux de l'exécution des tâches planifiées se trouvent dans le répertoire `/var/log/rsyslog/local/eole-schedule/`.

Comment changer le jour de mise à jour d'un serveur EOLE ?

Le jour tiré au hasard pour les mises à jour ne me convient pas et je souhaiterais le changer.

```
1 root@eole:~# manage_schedule -l
2 Tâches planifiées EOLE :
3 * les tâches hebdomadaires se feront le vendredi à 05:35 (hors sauvegarde)
4 - après sauvegarde
5 + Mise à jour du serveur (majauto)
6 root@eole:~#
```



Une solution consiste à supprimer le fichier de configuration `/etc/eole/extra/schedule/config.eol`.

```
1 root@eole:~# rm /etc/eole/extra/schedule/config.eol
2 rm : supprimer fichier '/etc/eole/extra/schedule/config.eol' ? y
3 root@eole:~# manage_schedule -l
4 Tâches planifiées EOLE :
5 * les tâches hebdomadaires se feront le jeudi à 04:12 (hors sauvegarde)
6 - après sauvegarde
7 + Mise à jour du serveur (majauto)
8 root@eole:~#
```

Chapitre 11

Gestion du pare-feu eole-firewall

Introduction

`eole-firewall` est conçu pour gérer les flux réseau d'un module EOLE.

Il permet d'autoriser des connexions :

- de l'extérieur vers le maître ;
- de l'extérieur vers un conteneur.

Techniquement, ces autorisations se traduisent par des règles *iptables* et, si nécessaire, des connexions TCP Wrapper^[p.64] et l'activation de modules noyau.



`eole-firewall` ne gère que des "autorisations", des règles en INPUT sur un port déterminé.

Les flux sont bloqués en entrée depuis l'extérieur. En interne (entre le maître et les conteneurs et entre conteneurs) il n'y a pas de restriction.

Si un conteneur possède une seconde interface (variable du type : *adresse_ip_link*), les flux sont bloqués en entrée.

eole-firewall avec ERA

Pour les modules avec ERA, Amon et AmonEcole, les règles d'`eole-firewall` ne s'appliquent pas. Seules les règles ERA du modèle choisi s'appliquent.

eole-firewall sans ERA

`eole-firewall` ne gère que des "autorisations", des règles en INPUT sur un port déterminé. Ces autorisations peuvent être affinées avec des "restrictions".



Les flux sont bloqués en entrée depuis l'extérieur. En interne (entre le maître et les conteneurs et entre conteneurs) il n'y a pas de restriction.

Si un conteneur possède une seconde interface (variable du type : *adresse_ip_link*), les flux sont bloqués en entrée.

Pour gérer les "autorisations" il faut créer des dictionnaires personnalisés. Pour cela il faut se référer à la rubrique traitant des dictionnaires dans la personnalisation du module à l'aide de Creole.

Pour des cas particuliers et exceptionnels il est possible de décrire des règles de pare-feu dans des fichiers placés dans le répertoire `/usr/share/eole/bastion/data/`.

Ces fichiers de règles doivent respecter les critères suivants :

- commencer par `#!/bin/bash` ;
- être exécutable ;
- ne pas contenir d'extension ;
- son code retour doit être 0.



La création de règles par cette méthode doit rester exceptionnelle.

Fichier `/usr/share/eole/bastion/data/40-icmp_static_rules` sur le module Scribe

```
1 #!/bin/bash
2 /sbin/iptables -A eth0-root -p icmp --icmp-type destination-unreachable -j
  ACCEPT
3 /sbin/iptables -A eth0-root -p icmp --icmp-type network-unreachable -j
  ACCEPT
4 /sbin/iptables -A eth0-root -p icmp --icmp-type source-quench -j ACCEPT
5 /sbin/iptables -A eth0-root -p icmp --icmp-type fragmentation-needed -j
  ACCEPT
6 /sbin/iptables -A eth0-root -p icmp --icmp-type time-exceeded -j ACCEPT
7 /sbin/iptables -A eth0-root -p icmp --icmp-type parameter-problem -j
  ACCEPT
8 /sbin/iptables -A eth0-root -p icmp --icmp-type echo-reply -j ACCEPT
9 /sbin/iptables -A eth0-root -p icmp --icmp-type echo-request -j ACCEPT
```

Créer des dictionnaires personnalisés pour gérer les règles du pare-feu eole-firewall

Utiliser des fichiers templates, paquets, services et règles de pare-feu ^[p.9]

Glossaire

Bareos	<p>Bareos est un ensemble de programmes qui permet de gérer les sauvegardes, les restaurations ou la vérifications de données d'un ordinateur sur un réseau hétérogène.</p> <p>En termes techniques, il s'agit d'un programme de sauvegarde client/serveur. Il est relativement facile d'utilisation et efficace. Il offre de nombreuses fonctions avancées de gestion de stockage qui facilitent la recherche et la restauration de fichiers perdus ou endommagés.</p>
Conteneur = <i>LXC</i>	<p>Un conteneur est une zone isolée à l'intérieur du système qui a un espace spécifique du système de fichiers, un réseau, des processus, des allocations mémoires et processeurs, comme s'il s'agissait de plusieurs serveurs physiques séparés.</p> <p>Contrairement à la virtualisation, une seule instance du noyau est présente pour l'ensemble des conteneurs et du maître.</p>
Creole = <i>Création EOLE</i>	<p>Creole gère la personnalisation des options de configuration des modules, le redémarrage des services, l'installation de paquets additionnels, la mise à jour du système.</p> <p>Il a été conçu pour être facilement personnalisable pour l'utilisateur final. Un ensemble d'outils est proposé pour modifier ou étendre les fonctionnalités offerte par EOLE.</p>
cron	<p>cron est un programme qui permet aux utilisateurs des systèmes Unix d'exécuter automatiquement des scripts, des commandes ou des logiciels à une date et une heure spécifiées à l'avance, ou selon un cycle défini à l'avance.</p>
Dictionnaire Creole	<p>Fichier, au format XML, décrivant l'ensemble de variables, de fichiers, de services et de paquets personnalisés en vue de configurer un serveur.</p>
DTD = <i>Document Type Definition</i>	<p>La Définition de Type de Document, est un document permettant de décrire un modèle de document SGML ou XML. Le modèle est décrit comme une grammaire de classe de documents : grammaire parce qu'il décrit la position des termes les uns par rapport aux autres, classe parce qu'il forme une généralisation d'un domaine particulier, et document parce qu'on peut former avec un texte complet.</p> <p>Une DTD décrit les documents à deux niveaux :</p> <ul style="list-style-type: none"> • la structure logique, que l'on peut assimiler à la syntaxe abstraite ; • la structure physique, que l'on peut assimiler à la syntaxe concrète. <p>Source : http://fr.wikipedia.org/wiki/Document_Type_Definition</p>

<p>Erlang</p>	<p>Erlang est un langage de programmation, supportant plusieurs paradigmes : concurrent, temps réel, distribué. Son cœur séquentiel est un langage fonctionnel à évaluation stricte, affectation unique, au typage dynamique fort. Sa couche concurrente est fondée sur le modèle d'acteur. Il possède des fonctionnalités de tolérance aux pannes et de mise à jour du code à chaud, permettant le développement d'applications à très haute disponibilité. Erlang est conçu pour s'exécuter sur une machine virtuelle spécifique appelée BEAM.</p> <p>Source Wikipédia : http://fr.wikipedia.org/wiki/Erlang_%28langage%29</p>
<p>Patch</p>	<p>Les modules EOLE sont livrés avec un ensemble de templates de fichiers de configuration qui seront copiés vers leur emplacement de destination à l'instance ou à chaque reconfigure.</p> <p>Il est possible de personnaliser ces fichiers de configuration à l'aide d'un patch.</p> <p>La procédure pour réaliser des patches est expliquée dans la rubrique Personnalisation du serveur à l'aide de Creole dans les documentations complètes ou dans la documentation partielle dédiée nommée PersonnalisationEOLEAvecCreole.</p>
<p>PID = <i>Process Identifier</i></p>	<p>L'identifiant de processus ou PID est un code unique attribué sur les systèmes Unix ou Windows à tout processus lors de son démarrage. Il permet ainsi d'identifier le processus dans la plupart des commandes s'appliquant sur un processus donné (comme kill).</p> <p>Wikipédia : https://fr.wikipedia.org/wiki/Identifiant_de_processus</p>
<p>TCP Wrapper = <i>tcpd</i></p>	<p>TCP Wrapper est une technique, propre à Unix, permettant de contrôler les accès à un service (ou démon) suivant la source.</p> <p>Il se configure grâce au deux fichiers <code>/etc/hosts.allow</code> et <code>/etc/hosts.deny</code>.</p> <p>Tous les démons ne supportent pas la technique TCP Wrapper.</p>
<p>Template = <i>Modèle Creole</i></p>	<p>Un template est un fichier contenant des variables Creole, qui sera instancié pour générer un fichier cible (typiquement un fichier de configuration serveur).</p>
<p>Tiramisu = <i>Outil de gestion de configuration</i></p>	<p>À cause de l'afflux de plus en plus grand des options de configuration des serveurs EOLE (plus de 1600 au dernier recensement), il était devenu de plus en plus difficile de correctement récupérer les options et de les utiliser là où elles devaient effectivement être employées. Pour remédier à ces difficultés, l'outil Tiramisu a été développé, il est utilisé comme moteur du générateur de configuration de la version EOLE 2.4.</p> <p>La documentation technique du projet : http://tiramisu.labs.libre-entreprise.org</p>

	<p>Les sources du projet Tiramisu : http://labs.libre-entreprise.org/projects/tiramisu/</p>
<p>Unicode</p>	<p>Unicode est un standard informatique qui permet des échanges de textes dans différentes langues, à un niveau mondial. Il est développé par le Consortium Unicode, qui vise à permettre le codage de texte écrit en donnant à tout caractère de n'importe quel système d'écriture un nom et un identifiant numérique, et ce de manière unifiée, quelle que soit la plate-forme informatique ou le logiciel.</p> <p>Source Wikipédia : http://fr.wikipedia.org/wiki/Unicode</p>
<p>XML = <i>Extensible Markup Language</i></p>	<p>L'Extensible Markup Language (« langage de balisage extensible » en français) est un langage informatique de balisage générique qui dérive du SGML. Cette syntaxe est dite « extensible » car elle permet de définir différents espaces de noms, c'est-à-dire des langages avec chacun leur vocabulaire et leur grammaire, comme XHTML, XSLT, RSS, SVG... Elle est reconnaissable par son usage des chevrons (< >) encadrant les balises. L'objectif initial est de faciliter l'échange automatisé de contenus complexes (arbres, texte riche...) entre systèmes d'informations hétérogènes (interopérabilité). Avec ses outils et langages associés une application XML respecte généralement certains principes :</p> <ul style="list-style-type: none"> • la structure d'un document XML est définie et validable par un schéma, • un document XML est entièrement transformable dans un autre document XML. <p>Source : http://fr.wikipedia.org/wiki/XML</p>