Fabric Controller
Installation Manual

Version 1.0

October. 2017

目次

改版履歴

版数	日付	変更内容
1.0	2017/10/20	初版制定

1. 概要

本マニュアルは、Fabric Controller (以下、FC)のインストール方法および冗長化設定について記載します。

1.1 関連マニュアル

FC をインストール、使用する際には、必要に応じて次の取り扱い説明書やオンライン情報を参照してください。

- Linux 『CentOS(http://www.centos.org/)』
- RDBMS 『PostgreSQL(http://www.postgresql.org/)』
- 冗長化 『Pacemaker(http://clusterlabs.org/)』

1.2 商標

本ドキュメントに記載されている会社名、商品名は、各社の登録商標または商標です。

2. 動作環境

2.1 FC の構成

FC の構成を図 2-1 に示します。

FC の構成は、青枠で囲まれた箇所です。

FC は、単体構成、冗長化構成のいずれでも動作させることができます。

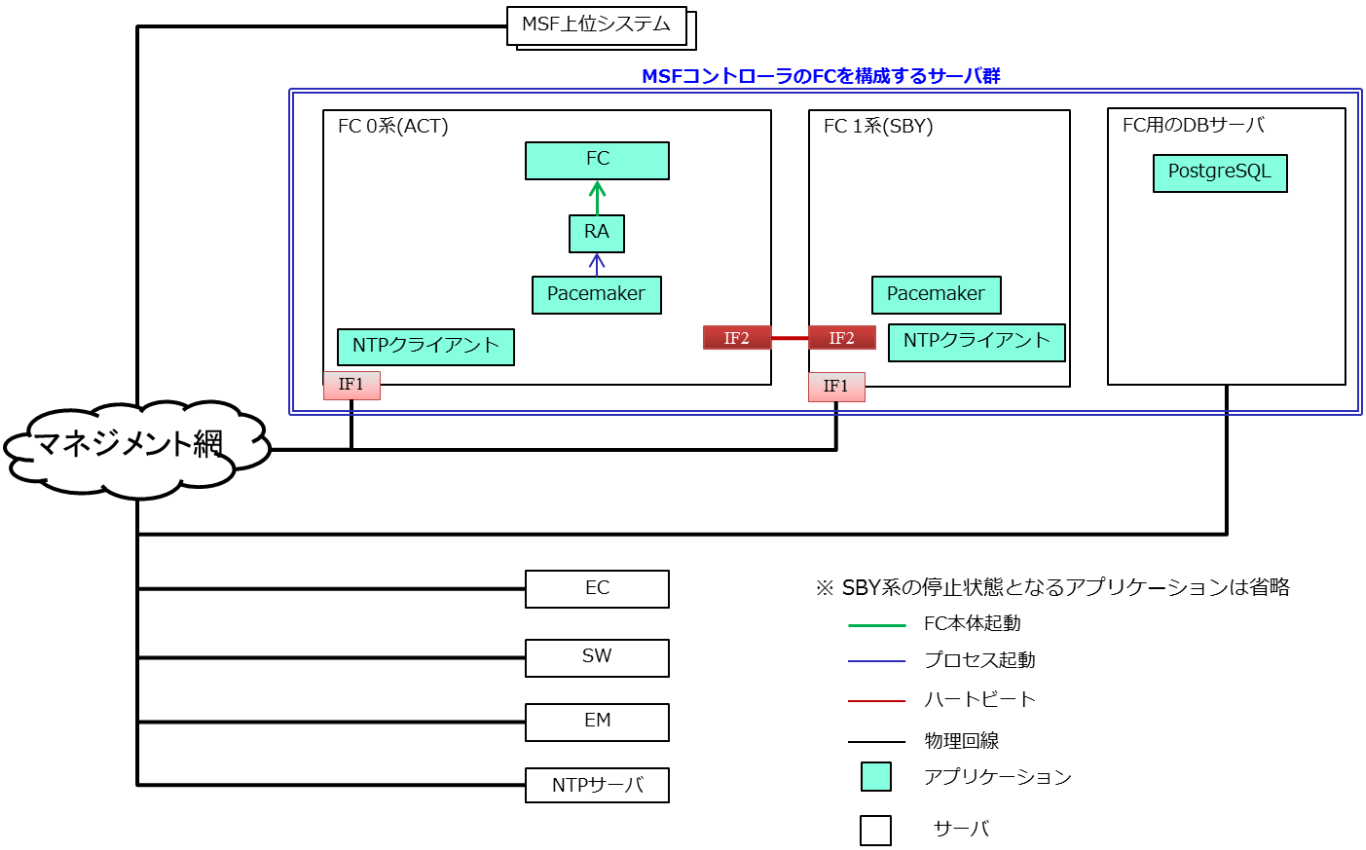

図 2-1 FC の構成

2.2 ハードウェア条件

FC は、x86 系 CPU 搭載 PC の Linux 上で動作します。表 2-1 に最低ハードウェアの条件を示します。

表 2-1 最低ハードウェア条件

No.	項目	スペック
1	CPU	2Core 以上
2	メモリ	1GB 以上
3	ハードディスク	空き容量 20GB 以上
4	NIC	1 ポート以上(FC を冗長化する場合は 2 ポート以上)

FC の動作確認済みハードウェア構成を参考として表 2-2 に示します。

表 2-2 動作確認済みハードウェア構成

No.	項目	スペック
1	CPU	Xeon E5-2420v2 @ 2.20GHz 6Core/12Thread
2	メモリ	32GB
3	ハードディスク	600GB
4	NIC	4 ポート

2.3 ソフトウェア条件

表 2-3 にソフトウェアの条件を示します。

各種 Java ライブラリについて、Java に含まれるものを除いて記載バージョンのソフトウェアをご使用ください。

表 2-3 ソフトウェア条件

No.	項目	ソフトウェア	バージョン	ソフトウェア入手先
1	OS	CentOS	7(1511)	http://www.centos.org/ ダウンロードファイル： CentOS-7-x86_64-DVD-1511.iso
2	ミドルウェア	Java	Oracle JDK 8 update101 以上 (Oracle JDK 9 以降は対象外)	http://www.oracle.com/ ダウンロードファイル： jdk-8u101-linux-x64.rpm
		Pacemaker	1.1.14-1	http://linux-ha.osdn.jp/wp/dl ダウンロードファイル： pacemaker-repo-1.1.14-1.1.el7.x86_64.rpm
		Corosync	2.3.5-1	Pacemaker のインストールパッケージに含まれる。
3	Java ライブラリ	Jetty	9.3.11	http://archive.eclipse.org/jetty/9.3.11.v20160721/dist/ ダウンロードファイル： jetty-distribution-9.3.11.v20160721.tar.gz
		Gson	2.7	https://repo1.maven.org/maven2/com/google/code/gson/gson/2.7/ ダウンロードファイル： gson-2.7.jar
		JAXB	2.2.8	Java に含まれる。
		Jersey	2.23.2	http://repo1.maven.org/maven2/org/glassfish/jersey/bundles/jaxrs-ri/2.23.2/ ダウンロードファイル： jaxrs-ri-2.23.2.tar.gz
		Hibernate	5.0.10	https://sourceforge.net/projects/hibernate/files/hibernate-orm/5.0.10.Final/hibernate-release-5.0.10.Final.tgz

No.	項目	ソフトウェア	バージョン	ソフトウェア入手先
				/download ダウンロードファイル： hibernate-release-5.0.10.Final.tgz
		SLF4J	1.6.1	https://sourceforge.net/projects/unirods/files/lib/slf4j-nop-1.6.1.jar/download?use_mirror=ayera&download=&failedmirror=kent.dl.sourceforge.net ダウンロードファイル： slf4j-nop-1.6.1.jar ※Hibernate 用追加ライブラリ
		Log4J	2.6.2	http://archive.apache.org/dist/logging/log4j/2.6.2 ダウンロードファイル： apache-log4j-2.6.2-bin.tar.gz
		Apache Commons IO	2.5	https://commons.apache.org/proper/commons-io/download_io.cgi ダウンロードファイル： commons-io-2.5-bin.tar.gz
		Hipster4j	1.0.1	http://www.hipster4j.org/ ダウンロードファイル： citiususc-hipster-v1.0.1-0-g09cf69c.zip
		JDBC	9.4.1209	https://jdbc.postgresql.org/download/ ダウンロードファイル： postgresql-9.4.1209.jar

3. FC のインストール方法

本マニュアルでは、図 2-1 の内、FC サーバ、FC 本体、FC 用の DB の設定について記載します。

3.1 各種ライブラリのインストール方法

本マニュアルでは、作業ユーザを一般ユーザ「msfctrl」として説明します。ご使用の環境に合わせてユーザを適宜置き換えてください。

3.1.1 OS の設定

(1) 作業ユーザを wheel グループに追加します。

```
$ su -
# gpasswd -a msfctrl wheel
# id msfctrl
----------------------------------------
uid=1000(msfctrl) gid=1000(msfctrl) groups=1000(msfctrl),10(wheel)
----------------------------------------
```

(2) visudo を実行して wheel グループが sudo コマンドを実行できるようにします。

```
# visudo
----------------------------------------
## Allows people in group wheel to run all commands
%wheel    ALL=(ALL)           ALL
----------------------------------------
# exit
```

※設定後、一度ログインし直します。

(3) SELinux を無効化します。

```
$ sudo vi /etc/sysconfig/selinux


----------------------------------------
SELINUX=disabled
----------------------------------------
```

(4) SELinux の設定を反映します。(OS 再起動)

```
$ sudo shutdown -r now
```

(5) ファイアウォールを停止します。

```
$ sudo systemctl disable firewalld.service
$ sudo systemctl stop firewalld.service
```

(6) NTP サーバと時刻同期をします。
※時刻同期は NTP クライアントソフトウェアを FC サーバにインストールして行います。

(7) unzip ツールをインストールします。
※Java ライブラリの提供形式が zip のものを展開するために必要です。

3.1.2 Java ライブラリの準備

ここでは、JDK のインストールと Java ライブラリの準備をします。

Java ライブラリは、「~/java_lib/」配下に準備することとします。

「~/java_lib/」配下のディレクトリ構成は以下のようにしてください。

/home/msfctrl	作業ユーザのホームディレクトリ
└-- java_lib	Java ライブラリ準備用ディレクトリ
├-- gson	Gson ライブラリ格納ディレクトリ
├-- hipster4j	Hipster4j ライブラリ格納ディレクトリ
├-- jetty	Jetty ライブラリ格納ディレクトリ
├-- postgresql	JDBC ライブラリ格納ディレクトリ
├-- apache-commons	Apache Commons IO ライブラリ格納ディレクトリ
├-- hibernate	Hibernate ライブラリ格納ディレクトリ
├-- jersey	Jersey ライブラリ格納ディレクトリ
└-- log4j	Log4J ライブラリ格納ディレクトリ

3.1.2.1 JDK のインストール

(1) http://www.oracle.com/から JDK の rpm ファイルをダウンロードします。
※以降の説明ではダウンロードした rpm ファイルを“jdk-8u101-linux-x64.rpm”とします。
※JDK のバージョンは、JDK8 update 101 以上のものをご使用ください。
ただし JDK9 以降は対象外です。

(2) ダウンロードした rpm ファイルを FC サーバの任意の場所に配置します。
ここでは、「~/rpm/」配下に rpm ファイルを配置することとします。

(3) rpm 配置ディレクトリに移動し、JDK をインストールします。

```
$ cd ~/rpm/
$ sudo rpm -ivh jdk-8u101-linux-x64.rpm
```

(4) 利用する Java を切り替えます。
※/usr/java/jdk1.8.0_101/jre/bin/java の番号を選択します。

```
$ sudo alternatives --config java
```

3.1.2.2 Gson ライブラリの準備

(1) https://repo1.maven.org/maven2/com/google/code/gson/gson/2.7/から
gson-2.7.jar をダウンロードします。

(2) ダウンロードした gson-2.7.jar を FC サーバの任意の場所に配置します。
ここでは、「~/download/」配下にファイルを配置することとします。

(3) gson-2.7.jar を「~/java_lib/gson/」配下に配置します。

```
$ cd ~/download/
$ mv gson-2.7.jar ~/java_lib/gson/
```

3.1.2.3 Hipster4j ライブラリの準備

Hipster4j は、Apache Maven を利用してソースからコンパイルします。
そのため Apache Maven をインストールします。

(1) https://maven.apache.org/download.cgi から
apache-maven-3.5.0-bin.tar.gz をダウンロードします。

(2) ダウンロードした apache-maven-3.5.0-bin.tar.gz を FC サーバの任意の場所に配置します。
ここでは、「~/download/」配下にファイルを配置することとします。

(3) apache-maven-3.5.0-bin.tar.gz を展開して、使用できるように環境変数を設定します。(ここでは、~/download/apache-maven-3.5.0 で環境を構築します。)

```
$ cd ~/download/
$ tar xvfz apache-maven-3.5.0-bin.tar.gz
$ cd ~/
$ vi .bash_profile
下記内容を追加。
-----
export M3_HOME=/home/msfctrl/download/apache-maven-3.5.0
M3=$M3_HOME/bin
export PATH=$M3:$PATH
---
```

※.bash_profile 編集後は、環境変数反映のため、ログアウト・再ログインをしてください。

(4) http://www.hipster4j.org/から
citiususc-hipster-v1.0.1-0-g09cf69c.zip をダウンロードします。

(5) ダウンロードした citiususc-hipster-v1.0.1-0-g09cf69c.zip を FC サーバの任意の場所に配置します。
ここでは、「~/download/」配下にファイルを配置することとします。

(6) citiususc-hipster-v1.0.1-0-g09cf69c.zip を展開し、展開したディレクトリ上で Apache Marven を用いて jar ファイルを生成します。生成した hipster-all-1.0.1-all.jar を「~/java_lib/hipster4j/」配下に

配置します。

```
$ cd ~/download/
$ unzip citiususc-hipster-v1.0.1-0-g09cf69c.zip
$ cd citiususc-hipster-09cf69c/
$ mvn package
$ cd ~/download/citiususc-hipster-09cf69c/hipster-all/target/
$ mv hipster-all-1.0.1-all.jar ~/java_lib/hipster4j/
```

3.1.2.4 Jetty ライブラリの準備

(1) http://archive.eclipse.org/jetty/9.3.11.v20160721/dist/ から
jetty-distribution-9.3.11.v20160721.tar.gz をダウンロードします。

(2) ダウンロードした jetty-distribution-9.3.11.v20160721.tar.gz を FC サーバの任意の場所に配置します。
ここでは、「~/download/」配下にファイルを配置することとします。

(3) jetty-distribution-9.3.11.v20160721.tar.gz を展開し、展開したディレクトリ内の lib ディレクトリ一式を「~/java_lib/jetty/」配下に配置します。

```
$ cd ~/download/
$ tar xvfz jetty-distribution-9.3.11.v20160721.tar.gz
$ cd jetty-distribution-9.3.11.v20160721/
$ mv lib ~/java_lib/jetty/
```

3.1.2.5 JDBC ライブラリの準備

(1) https://jdbc.postgresql.org/download/postgresql-9.4-1209.jar から
postgresql-9.4.1209.jar をダウンロードします。

(2) ダウンロードした postgresql-9.4.1209.jar を FC サーバの任意の場所に配置します。
ここでは、「~/download/」配下にファイルを配置することとします。

(3) postgresql-9.4.1209.jar を「~/java_lib/postgresql」配下に配置します。

```
$ cd ~/download/
$ mv postgresql-9.4.1209.jar ~/java_lib/postgresql/
```

3.1.2.6 Apache Commons IO ライブラリの準備

(1) https://commons.apache.org/proper/commons-io/download_io.cgi から
commons-io-2.5-bin.tar.gz をダウンロードします。

(2) ダウンロードした commons-io-2.5-bin.tar.gz を FC サーバの任意の場所に配置します。
ここでは、「~/download/」配下にファイルを配置することとします。

(3) commons-io-2.5-bin.tar.gz を展開し、展開したディレクトリ内の commons-io-2.5.jar を「~/java_lib/apache-commons/」配下に配置します。

```
$ cd ~/download/
$ tar xvfz commons-io-2.5-bin.tar.gz
$ cd commons-io-2.5/
$ mv commons-io-2.5.jar ~/java_lib/apache-commons/
```

3.1.2.7 Hibernate ライブラリの準備

Hibernate のライブラリは本体と追加ライブラリを準備します。

(1) https://sourceforge.net/projects/hibernate/files/hibernate-orm/5.0.10.Final/hibernate-release-5.0.10.Final.tgz/download から
hibernate-release-5.0.10.Final.tgz をダウンロードします。

(2) ダウンロードした hibernate-release-5.0.10.Final.tgz を FC サーバの任意の場所に配置します。
ここでは、「~/download/」配下にファイルを配置することとします。

(3) hibernate-release-5.0.10.Final.tgz を展開し、展開したディレクトリ内の lib ディレクトリ一式を「~/java_lib/hibernate/」配下に配置します。

```
$ cd ~/download/
$ tar xvfz hibernate-release-5.0.10.Final.tgz
$ cd hibernate-release-5.0.10.Final/
$ mv lib ~/java_lib/hibernate/
```

(4) Hibernate に必要なライブラリを追加します。
https://sourceforge.net/projects/unirods/files/lib/slf4j-nop-1.6.1.jar/download?use_mirror=ayera&download=&failedmirror=kent.dl.sourceforge.net から
slf4j-nop-1.6.1.jar をダウンロードします。

(5) ダウンロードしたファイルを FC サーバの任意の場所に配置します。
ここでは、「~/download/」配下にファイルを配置することとします。

(6) slf4j-nop-1.6.1.jar を「~/java_lib/hibernate/lib/ optional/ehcache」配下に配置します。

```
$ cd ~/download/
$ mv slf4j-nop-1.6.1.jar ~/java_lib/hibernate/lib/optional/ehcache/
```

3.1.2.8 Jersey ライブラリの準備

(1) http://repo1.maven.org/maven2/org/glassfish/jersey/bundles/jaxrs-ri/2.23.2/から jaxrs-ri-2.23.2.tar.gz をダウンロードします。

(2) ダウンロードした jaxrs-ri-2.23.2.tar.gz を FC サーバの任意の場所に配置します。
ここでは、「~/download/」配下にファイルを配置することとします。

(3) jaxrs-ri-2.23.2.tar.gz を展開して、展開したディレクトリ内のファイルおよびディレクトリ一式を「~/java_lib/jersey/」配下に配置します。

```
$ cd ~/download/
$ tar xvfz jaxrs-ri-2.23.2.tar.gz
$ cd jaxrs-ri/
$ mv * ~/java_lib/jersey/
```

3.1.2.9 Log4J ライブラリの準備

(1) http://archive.apache.org/dist/logging/log4j/2.6.2 から apache-log4j-2.6.2-bin.tar.gz をダウンロードします。

(2) ダウンロードした apache-log4j-2.6.2-bin.tar.gz を FC サーバの任意の場所に配置します。
ここでは、「~/download/」配下にファイルを配置することとします。

(3) apache-log4j-2.6.2-bin.tar.gz を展開し、展開したディレクトリ内の log4j-1.2-api-2.6.2.jar、log4j-api-2.6.2.jar、log4j-core-2.6.2.jar を「~/java_lib/log4j/」配下に配置します。

```
$ cd ~/download/
$ tar xvfz apache-log4j-2.6.2-bin.tar.gz
$ cd apache-log4j-2.6.2-bin/
$ mv log4j-1.2-api-2.6.2.jar ~/java_lib/log4j/
$ mv log4j-api-2.6.2.jar ~/java_lib/log4j/
$ mv log4j-core-2.6.2.jar ~/java_lib/log4j/
```

3.2 FC のインストール

FC のインストール方法を示します。

3.2.1 FC サーバ内ディレクトリ構成

FC のディレクトリ構成(主要ファイルおよびライブラリ格納ディレクトリ名)を示します。

本マニュアルでは、インストール用ディレクトリを「~/msf-controller/」とします。

/home/msfctrl	作業ユーザのホームディレクトリ
└-- msf-controller	インストール用ディレクトリ
｜-- bin	FC 起動停止用スクリプト格納ディレクトリ
｜　├--fc_ctl.sh	FC 起動停止用スクリプト本体
｜　└--fc	FC 用リソースエージェント
├-- lib	FC 本体およびライブラリ格納ディレクトリ
｜　├-- FabricController.jar	FC 本体
｜　├-- gson	Gson ライブラリ格納ディレクトリ
｜　├-- hipster4j	Hipster4j ライブラリ格納ディレクトリ
｜　├-- jetty	Jetty ライブラリ格納ディレクトリ
｜　├-- postgresql	JDBC ライブラリ格納ディレクトリ
｜　├-- apache-commons	Apache Commons IO ライブラリ格納ディレクトリ
｜　├-- hibernate	Hibernate ライブラリ格納ディレクトリ
｜　├-- jersey	Jersey ライブラリ格納ディレクトリ
｜　└-- log4j	Log4J ライブラリ格納ディレクトリ
├-- logs	ログディレクトリ※1
└--conf	コンフィグディレクトリ
├-- fc_system.xml	FC システム設定コンフィグ
├-- fc_data.xml	FC 初期設定コンフィグ
├-- fc_develop.xml	FC 内部動作設定コンフィグ
├-- log4j2.xml	Log4J コンフィグ
└-- hibernate.cfg.xml	Hibernate コンフィグ

※1. ログディレクトリは FC 起動時点で自動生成されます。FC インストール時点は存在しません。

3.2.2 FC 本体のインストール

(1) FC の格納 tar ファイル(msf-controller.tar.gz)をインストール用ディレクトリに配置します。

(2) 配置した FC の格納 tar ファイルを展開します。
※再インストールは、tar ファイルから展開されたディレクトリを一括削除した後、新しい tar ファイルを展開してください。

```
$ cd ~/msf-controller/
$ tar xvfz ~/msf-controller/msf-controller.tar.gz
```

(3) Java ライブラリを FC の「lib」配下にコピーします。
※Java ライブラリを準備したディレクトリは FC 再インストールに備えて残置します。

```
$ cp -r ~/java_lib/* ~/msf-controller/lib/
```

(4) 展開したディレクトリ内に含まれるスクリプトファイル(fc_ctl.sh)に実行権を設定します。

```
$ cd ~/msf-controller/bin/
$ chmod 755 fc_ctl.sh
```

3.3 FC の設定

FC の設定について記載します。

3.3.1 初期データ準備

FC を初期化するにあたり、DB に登録する初期データを準備します。DB に初期登録するデータをまとめたコンフィグは FC 初期設定コンフィグです。FC 初期設定コンフィグの変更は基本的に FC の初回起動時または FC を初期化したい時のみです。FC 初期設定コンフィグの編集は FC 停止中に行います。FC 起動中の FC 初期設定コンフィグ編集結果は、FC の動作へ反映されません。

3.3.1.1 FC 初期設定コンフィグ 概要

FC 初期設定コンフィグの概要を表 3-1 に示します。

表 3-1 FC 初期設定コンフィグ概要

No.	コンフィグファイル名	コンフィグ名	配置場所
1	fc_data.xml	FC 初期設定コンフィグ	「FC インストールディレクトリ/conf/」配下

3.3.1.2 FC 初期設定コンフィグ 内容

FC 初期設定コンフィグの各パラメータについて、記載します。

FC 初期設定コンフィグファイルは xml 形式です。

下記は、FC 初期設定コンフィグのサンプルです。

FC 初期設定コンフィグの各パラメータについて、表 3-2 にまとめます。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dataConf xmlns="http://fc.msf/common/config/type/data">
    <swClustersData>
        <swClusterData>
            <swCluster>
                <swClusterId>1</swClusterId>
                <maxLeafNum>45</maxLeafNum>
                <maxSpineNum>4</maxSpineNum>
                <asNum>65000</asNum>
                <rpLoopbackAddress>1.1.1.1</rpLoopbackAddress>
                <interfaceStartAddress>1.1.1.1</interfaceStartAddress>
                <loopbackStartAddress>1.1.1.1</loopbackStartAddress>
                <managementStartAddress>1.1.1.1</managementStartAddress>
                <managementAddressPrefix>24</managementAddressPrefix>
            </swCluster>
            <rrs>
                <rr>
                    <rrNodeId>1</rrNodeId>
                    <rrRouterId>1.1.1.1</rrRouterId>
                </rr>
            </rrs>
        </swClusterData>
    </swClustersData>
    <slice>
        <ipv4MulticastAddressBase>239.0.1.0</ipv4MulticastAddressBase>
    </slice>
</dataConf>
```

表 3-2 FC 初期設定コンフィグ パラメーター一覧

要素	型	設定可能範囲	要素数	説明
dataConf	-	-	1	-
swClustersData	-	-	1	-
swClusterData	-	-	1	1 要素につき、1SW クラスタ分の情報に相当
swCluster	-	-	1	SW クラスタの情報
swClusterId	整数	1～100	1	SW クラスタ ID
maxLeafNum	整数	1～1000	1	SW クラスタの最大 Leaf 数[台]
maxSpineNum	整数	1～1000	1	SW クラスタの最大 Spine 数[台]
asNum	整数	0～65535	1	SW クラスタの AS 番号
rpLoopbackAddress	文字列	-	1	SW クラスタの RP ループバックアドレス(IPv4 形式のみ)
interfaceStartAddress	文字列	-	1	SW クラスタのインタフェースアドレス始点 IP アドレス(IPv4 形式のみ) 払い出すインタフェースアドレスのプレフィックスは 30 固定 SW クラスタのインタフェースアドレスは FC 上で管理する装置(Leaf、Spine)の内部リンク IF に設定する IP アドレス 始点 IP アドレスから始まり、インタフェース（装置）が追加される毎に、FC 内部で自動的に適当な IP アドレスを計算して、IP アドレスが割り振られる。
loopbackStartAddress	文字列	-	1	SW クラスタのループバックアドレス始点 IP アドレス(IPv4 形式のみ) 払い出すループバックアドレスのプレフィックスは 32 固定 SW クラスタのループバックアドレスは FC 上で管理する装置(Leaf、Spine)のループバックアドレス 始点 IP アドレスから始まり、インタフェース（装置）が追加される毎に、FC 内部で自動的に適当な IP アドレ

要素				型	設定可能範囲	要素数	説明
							スを計算して、IP アドレスが割り振られる。
			managementStartAddress	文字列	-	1	SW クラスタのマネジメント始点 IP アドレス マネジメント IP アドレスは MSF 上で管理する装置のマネジメント IF の IP アドレス 始点 IP アドレスから始まり、インタフェース（装置）が追加される毎に、FC 内部で自動的に適当な IP アドレスを計算して、IP アドレスが割り振られる。
			managementAddressPrefix	整数	1～32	1	SW クラスタのマネジメントアドレス向けプレフィックス
		rrs		-	-	0 以上	DB の RR 情報の 1 レコードに相当
			rr	-	-	1	-
			rrNodeId	整数	1～65535	1	SW クラスタの RR 装置 ID
			rrRouterId	文字列	-	1	SW クラスタの RR ルータ ID(IPv4 形式のみ)
	slice			-	-	1	-
		ipv4MulticastAddressBase		文字列	-	1	L2VPN 向けの IPv4 マルチキャストアドレスの基底値(IPv4 形式のみ)

3.3.2 運用時の設定

FC の運用時の設定について記載します。設定は、FC 停止中に FC システム設定コンフィグを編集して行います。FC 起動中の FC システム設定コンフィグ編集結果は、FC の動作へ反映されません。

3.3.2.1 FC システム設定コンフィグ 概要

FC システム設定コンフィグの概要を表 3-3 に示します。

表 3-3 システム設定コンフィグ概要

No.	コンフィグファイル名	コンフィグ名	配置場所
1	fc_system.xml	FC システム設定コンフィグ	「FC インストールディレクトリ/conf/」配下

3.3.2.2 FC システム設定コンフィグ 内容

FC システム設定コンフィグの各パラメータについて、記載します。

FC システム設定コンフィグファイルは xml 形式です。

下記は FC システム設定コンフィグのサンプルです。

FC システム設定コンフィグの各パラメータについて、表 3-4 にまとめます。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<systemConf xmlns="http://fc.msf/common/config/type/system">
    <rest>
        <server>
            <listeningAddress>0.0.0.0</listeningAddress>
            <listeningPort>18080</listeningPort>
        </server>
        <client>
            <waitConnectionTimeout>30</waitConnectionTimeout>
            <requestTimeout>600</requestTimeout>
        </client>
        <json>
            <isPrettyPrinting>true</isPrettyPrinting>
            <isSerializeNulls>false</isSerializeNulls>
        </json>
    </rest>
    <slice>
        <l2SlicesMagnificationNum>1</l2SlicesMagnificationNum>
        <l3SlicesMagnificationNum>1</l3SlicesMagnificationNum>
        <l2MaxSlicesNum>100</l2MaxSlicesNum>
        <l3MaxSlicesNum>100</l3MaxSlicesNum>
    </slice>
    <swClustersData>
        <swClusterData>
            <swCluster>
                <swClusterId>1</swClusterId>
                <ecControlAddress>0.0.0.0</ecControlAddress>
                <ecControlPort>18080</ecControlPort>
            </swCluster>
        </swClusterData>
    </swClustersData>
    <traffic>
        <interval>0</interval>
        <dataRetentionPeriod>7</dataRetentionPeriod>
        <tmToolPath>/opt/fc/tm/</tmToolPath>
        <tmInputFilePath>/opt/fc/tm/input/</tmInputFilePath>
        <tmOutputFilePath>/opt/fc/tm/output/</tmOutputFilePath>
    </traffic>
</systemConf>
```

表 3-4FC システム設定コンフィグ パラメーター一覧

要素	型	設定可能範囲※	要素数	説明
systemConf	-	-	1	-
rest	-	-	1	-
server	-	-	1	-
listeningAddress	文字列	-	1	REST 待ち受けインタフェースアドレス
listeningPort	整数	0～65535	1	REST 待ち受けポート
client	-	-	1	-
waitConnectionTimeout	整数	0 以上	1	REST 要求通信確立待ち時間[秒]
requestTimeout	整数	0 以上	1	REST 要求制御待ち時間[秒]
json	-	-	1	-
isPrettyPrinting	boolean	true/false	1	REST 要求に対して FC が応答する JSON データを改行/インデント付与など整形する(true)か否(false)か
isSerializeNulls	boolean	true/false	1	REST 要求に対して FC が応答する JSON データに値が null 値のものが存在した場合、パラメータを null 値で表示する(true)か否(false)か
slice	-	-	1	-
l2SlicesMagnificationNum	整数	1,2	1	L2 スライス数猶予倍率。払い出せるスライス ID の最大値を制御する値。 l2MaxSlicesNum の値をもとに FC が払い出す L2 スライス ID の範囲を決定する。
l3SlicesMagnificationNum	整数	1,2	1	L3 スライス数猶予倍率。払い出せるスライス ID の最大値を制御する値。 l3MaxSlicesNum の値をもとに FC が払い出す L3 スライス ID の範囲を決定する。

要素	型	設定可能範囲※	要素数	説明
l2MaxSlicesNum	整数	1～1000	1	最大 L2 スライス数
l3MaxSlicesNum	整数	1～1000	1	最大 L3 スライス数
swClustresData	-	-	1	-
swClusterData	-	-	1	-
swCluster	-	-	1	-
swClusterId	整数	1～100	1	SW クラスタ ID
ecControlAddress	文字列	-	1	EC のアドレス
ecControlPort	整数	0～65535	1	EC のポート番号
traffic	-	-	1	本バージョンでは動作対象外
interval	整数	0 固定	1	-
dataRetentionPeriod	整数	-	1	-
tmToolPath	文字列	-	1	-
tmInputFilePath	文字列	-	1	-
tmOutputFilePath	文字列	-	1	-

※ 設定範囲の上限値が指定されていない属性の上限値は、int 型最大値(2147483647)です。

3.3.3 DB の設定

DB の初期設定を下記に示します。設定は、FC 用の DB サーバ上で実施してください。事前に FC 用の DB サーバに PostgreSQL がインストールされ、「postgres」ユーザが存在することを前提とします。

※FC サーバと FC 用の DB サーバは別ホストです。

※FC 用の DB サーバの作業ユーザは任意とします。

※本節は FC 用の DB サーバ上での操作を記載します。

(1) データベースクラスタを初期化します。

```
$ sudo su - postgres
$ initdb --no-locale -E UTF-8
$ exit
```

(2) PostgreSQL を起動します。

```
$ sudo systemctl start postgresql-9.3.service
```

(3) ロール、データベースを作成します。

※ここで指定するロール名、ロールパスワードは FC の hibernate コンフィグに設定をします。

Hibernate コンフィグ(hibernate.cfg.xml)の設定は hibernate の公式ページを参考に設定してください。

```
$ sudo -u postgres psql
postgres=# create role ロール名 with login password 'ロールパスワード';
postgres=# create database msf_fc LC_COLLATE 'en_US.UTF-8' LC_CTYPE 'en_US.UTF-8'
TEMPLATE template0;
postgres=# ¥q
```

(4) テーブルおよびテーブルスキーマを作成します。

※「3.2.2 」で展開した tar ファイル内に含まれる”msf_fc.sql”を予め FC 用の DB サーバの「~/」配下に配置します。

```
$ cd ~/
$ sudo -u postgres psql -U ロール名 -d msf_fc < msf_fc.sql
```

3.3.4 FC 簡易起動確認

FC 単体構成の簡易起動確認手順を以下に記載します。

FC 冗長化構成の場合の FC 起動停止は Pacemaker を用いて行います。冗長化構成の設定は「4.」を、Pacemaker については公式ページを参照ください。

(1) FC のコンフィグ(FC 初期設定コンフィグ、FC システム設定コンフィグ、Hibernate コンフィグ)を適宜変更します。

(2) FC 起動停止スクリプトの配置場所へ移動し、以下のコマンドを実行します。

```
$ sh fc_ctl.sh start
```

FC が正常に起動した場合と起動失敗した場合のプロンプトの状態について記載します。

FC が起動した場合のプロンプトの状態
(何も表示されずにプロンプト"$"が表示されます)

```
$
```

FC が起動失敗した場合のプロンプトの状態

```
ERROR. hogehoge
$
```

※hogehoge は FC を起動できなかった理由または状態を表示します。
FC が正常に起動しなかった場合は、コンフィグやネットワークの設定が正しくない可能性があります。

(3) FC を停止します。
FC 起動停止スクリプトの配置場所へ移動し、以下のコマンドを実行します。

```
$ sh fc_ctl.sh stop
```

FC が正常に停止した場合と停止失敗した場合のプロンプトの状態を記載します。

FC が停止した場合のプロンプトの状態

```
INFO. FabricController stopped.
$
```

FC が停止失敗した場合のプロンプトの状態

```
ERROR. hogehoge
$
```

※hogehoge は FC を停止できなかった理由または状態を表示します。

3.3.5 FC 起動停止スクリプトによる状態確認

FC 起動停止スクリプトによる状態確認手順を記載します。

確認できる内容は、FC の起動状態(プロセス有無)です。

FC 起動停止スクリプトの配置場所へ移動し、以下のコマンドを実行します。

```
$ sh fc_ctl.sh status
```

FC の起動状態は、上記コマンドの結果(標準出力ログ)から判断します。標準出力ログについて、表 3-5 にまとめます。

表 3-5 FC の起動状態

FC の起動状態	標準出力ログ	備考
起動中(正常)	INFO. FabricController[pid=PID] is running.	PID は、FC の実際の pid が表示される。
起動中(異常:二重起動状態)	WARN. FabricController is running(two or more applications are running).	2 つ以上 FC が起動している。正常動作に戻すため、誤って起動した FC を停止させる。 この場合、FC 起動停止スクリプトで FC を停止することができないため、kill コマンドを用いて FC を停止する必要がある。
停止中	INFO. FabricController is not running.	-

3.3.6 FC の状態判断(正常/異常)

FC の状態について確認する手段を記載します。

FC の状態を確認するには、表 3-6 の REST メッセージを FC のマネジメントアドレスに送信します。

表 3-6 状態確認 REST リクエスト

メソッド	URI
GET	/v1/MSFcontroller/status

FC の状態は、送信した REST のレスポンスコードを確認することで判断できます。

ただし、レスポンスが無い場合は FC が停止しているか、通信異常の状態です。

レスポンスのコード一覧を表 3-7 にまとめます。

表 3-7 状態確認レスポンスコード一覧

レスポンスコード	FC の状態
200	正常
500	異常

また、レスポンスコードが 200 の場合、当該レスポンスの body 部に載る情報を参考に表 3-8 に記載します。

表 3-8 状態確認レスポンス body 部

body パラメータ名	body パラメータ概要	body パラメータ値	body パラメータ値概要
service_status	サービス起動状態	start-up in progress	起動準備中
		running	起動
		shutdown in progress	停止準備中
		system switching	系切替中
blockade_status	保守閉塞状態	blockade	閉塞中
		none	閉塞なし

4. 冗長化設定

FC の冗長化は、Pacemaker を使用します。ここでは、FC の冗長化設定に関して記載します。Pacemaker のインストールおよび基本的な設定は、Pacemaker の公式ページを参考にして実施してください。

4.1 リソースエージェントの配置

FC 用リソースエージェントを所定のディレクトリに配置します。

FC 用リソースエージェントは、FC の提供バイナリ内に存在します。

FC 用リソースエージェントのファイル名は、”fc”です。

(1) 0 系ノード、1 系ノードにて、FC 用リソースエージェントを配置します。(root 権限で実施します。)

```
# cp -p ~/msf-controller/bin/fc /usr/lib/ocf/resource.d/heartbeat/fc
```

(2) 0 系ノード、1 系ノードにて、FC 用リソースエージェントファイルに実行権、所有権(root)を設定します。

```
# cd /usr/lib/ocf/resource.d/heartbeat/
# chmod 755 fc
# chown root:root fc
```

4.2 Pacemaker の設定

Pacemaker の設定について、参考設定を記載します。

4.2.1 クラスタのプロパティ設定

クラスタのプロパティ設定について表 4-1 に示します。

FC は STONITH 機能を無効とします。

表 4-1 クラスタプロパティ設定

項目	値	概要	備考
no-quorum-policy	Ignore	ノード数によるリソース割当て	-
stonith-enabled	false	障害ノード対処(STONITH 制御)	STONITH を利用しない

4.2.2 リソースのデフォルト設定

リソースのデフォルト設定について表 4-2 に示します。

表 4-2 リソースデフォルト設定

項目	値	概要	備考
resource-stickiness	INFINITY	リソース割当て	自動フェイルバックなし
migration-threshold	1	リソース故障可能回数	1 回故障発生でフェイルオーバ

4.2.3 リソースの構成設定

リソースの構成設定について表 4-3 に示します。

各リソースエージェントの設定は、参考設定を「4.2.4 」、「4.2.5 」、「4.2.6 」、「4.2.7 」に示します。

表 4-3 リソース構成設定

リソースエージェント	リソース種別	概要	グループ設定有無	グループ内起動順番(停止は逆順)
VIPcheck	Primitive	仮想 IP チェック	有	1
FC 用リソースエージェント	Primitive	FC プロセス監視		2
IPaddr2	Primitive	仮想 IP 割り当て		3
diskd	Clone	内蔵 Disk 監視	無	-

4.2.4 VIPcheck の設定

VIPcheck の OCF パラメータの設定例を表 4-4 に示します。

表 4-4 VIPcheck の OCF パラメータ設定例

OCF パラメータ	概要	設定例
target_ip	チェックする仮想 IP	192.168.10.100
count	チェック回数	1
wait	待機時間(秒)	10

VIPcheck のオペレーション設定例を表 4-5 に示します。

表 4-5 VIPcheck のオペレーション設定例

オペレーション	タイムアウト値(秒)	監視間隔(秒)	障害時の動作	始動遅延時間(秒)
start	90	0	restart	6

4.2.5 FC 用リソースエージェントの設定

FC 用リソースエージェントの OCF パラメータについて概要を含めて設定例を表 4-6 に示します。

表 4-6 FC 用リソースエージェントの OCF パラメータ概要と設定例

OCF パラメータ	概要	設定例
host_0	FC の 0 系のホスト名	FC1-1
host_1	FC の 1 系のホスト名	FC1-2
fc_system_xml	FC システム設定コンフィグ配置場所	/home/msfctrl/msf-controller

OCF パラメータ	概要	設定例
		/conf/fc_system.xml
fc_ctl	FC 起動停止スクリプト配置場所	/home/msfctrl/msf-controller /bin/fc_ctl.sh
fc_username	FC のユーザ名	msfctrl

FC 用リソースエージェントのオペレーション設定例をに示します。

表 4-7 FC 用リソースエージェントのオペレーション設定例

オペレーション	タイムアウト値(秒)	監視間隔(秒)	障害時の動作	始動遅延時間(秒)
start	60	0	restart	(設定なし)
monitor	60	60	restart	(設定なし)
stop	3600	0	block	(設定なし)

4.2.6 IPaddr2 の設定

IPaddr2 の OCF パラメータの設定例を表 4-8 に示します。

表 4-8 IPaddr2 の OCF パラメータ設定例

OCF パラメータ	概要	設定例
ip	仮想 IP アドレス	192.168.10.100
nic	仮想 IP アドレスを設定する IF	enp0s99
cidr_netmask	仮想 IP アドレスのサブネットマスク	24

IPaddr2 のオペレーション設定例を表 4-9 に示します。

表 4-9 IPaddr2 のオペレーション設定例

オペレーション	タイムアウト値(秒)	監視間隔(秒)	障害時の動作	始動遅延時間(秒)
start	60	0	restart	(設定なし)
monitor	60	10	restart	(設定なし)
stop	60	0	ignore	(設定なし)

4.2.7 diskd の設定

diskd の OCF パラメータの設定例を表 4-10 に示します。

表 4-10 diskd の OCF パラメータ設定例

OCF パラメータ	概要	設定例
name	監視対象名	diskcheck_status_internal

OCF パラメータ	概要	設定例
device	監視対象のディスク名	/dev/sda1
interval	監視間隔(秒)	10

diskd のオペレーション設定例を表 4-11 に示します。

表 4-11 diskd のオペレーション設定例

オペレーション	タイムアウト値(秒)	監視間隔(秒)	障害時の動作	始動遅延時間(秒)
start	60	0	restart	(設定なし)
monitor	60	10	restart	(設定なし)
stop	60	0	ignore	(設定なし)

4.2.8 各種制約設定

本マニュアルでは、表 4-3 で示したリソース構成設定の内グループ化しているリソース ID を grpFC、Clone リソースを clnDiskd としてリソース制約設定を記載します。

設定は、Pacemaker 公式の「pm_crmgen_env.xls」ファイルをベースに記載します。

4.2.8.1 リソース配置制約設定

リソース配置制約の設定例を表 4-12 リソース配置制約設定例に示します。

表 4-12 リソース配置制約設定例

リソース ID	スコア	bool op (and/or)	条件属性名	条件	条件値	役割
grpFC	-inf	or	diskcheck_status_internal	not_defined		(設定なし)
				eq	ERROR	(設定なし)

4.2.8.2 リソース同居制約設定

リソース同居制約の設定例を表 4-13 に示します。

表 4-13 リソース同居制約設定例

制御関連リソース ID	制御対象リソース ID	スコア(重みづけ)	制約関連リソースの役割	制約対象リソースの役割
grpFC	clnDiskd	inf	(設定なし)	(設定なし)

4.2.8.3 リソース起動順序制約設定

リソース起動順序制約の設定例を表 4-14 に示します。

表 4-14 リソース起動順序制約設定例

先発リソース ID	後発リソース ID	スコア (重みづけ)	先発リソースのアクション	後発リソースのアクション	停止順序
clnDiskd	grpFC	0	(設定なし)	(設定なし)	起動と同順