Ksocket

リリース *v2.0.0*

Fixpoint, Inc.

2020 年 04 月 01 日

目次

Ksocket は 株式会社フィックスポイント[*1] （以下 Fixpoint）が開発を行っている、ネットワーク内の様々な情報を調査・収集するためのソフトウェアです。Kompira cloud[*2] と連携して動作する前提となっており、Kompira cloud からの実行命令を受けてネットワーク内の情報を収集します。

このドキュメントは Ksocket v2 系を対象としています。Ksocket v1 系のドキュメントをお探しの場合は Ksocket ダウンロードページ[*3] のアーカイブからダウンロードしてください。

[*1] https://www.fixpoint.co.jp/
[*2] https://cloud.kompira.jp/
[*3] https://blog.cloud.kompira.jp/entry/downloads

第 1 章

インストール

1.1 ローカルネットワークへのインストール

社内ネットワーク等のローカルネットワークをスキャンする Ksocket のインストール方法です。Ksocket ダウンロードページ[*4] から対象プラットフォーム用のインストーラをダウンロードし、各プラットフォームごとの指示に従ってください。

1.1.1 Linux

ダウンロードしてきたインストーラ（例: `ksocket-2.0.0.0-linux-x86_64.tar.gz` ）に対して **管理者権限** にて以下のように実行してください:

```
% tar xf ksocket-2.0.0.0-linux-x86_64.tar.gz
% ./ksinstall
```

上記によりインストーラーが起動するため *Ksocket* **インストーラー** に従ってインストールを進めてください。

[*4] https://blog.cloud.kompira.jp/entry/downloads

1.1.2 Windows

Ksocket は Windows にてパケットを扱うために Npcap[*5] が提供する API に依存しています。その為、事前に Npcap (0.9989 以上) を **WinPcap API-compatible mode** にてインストールしてください。

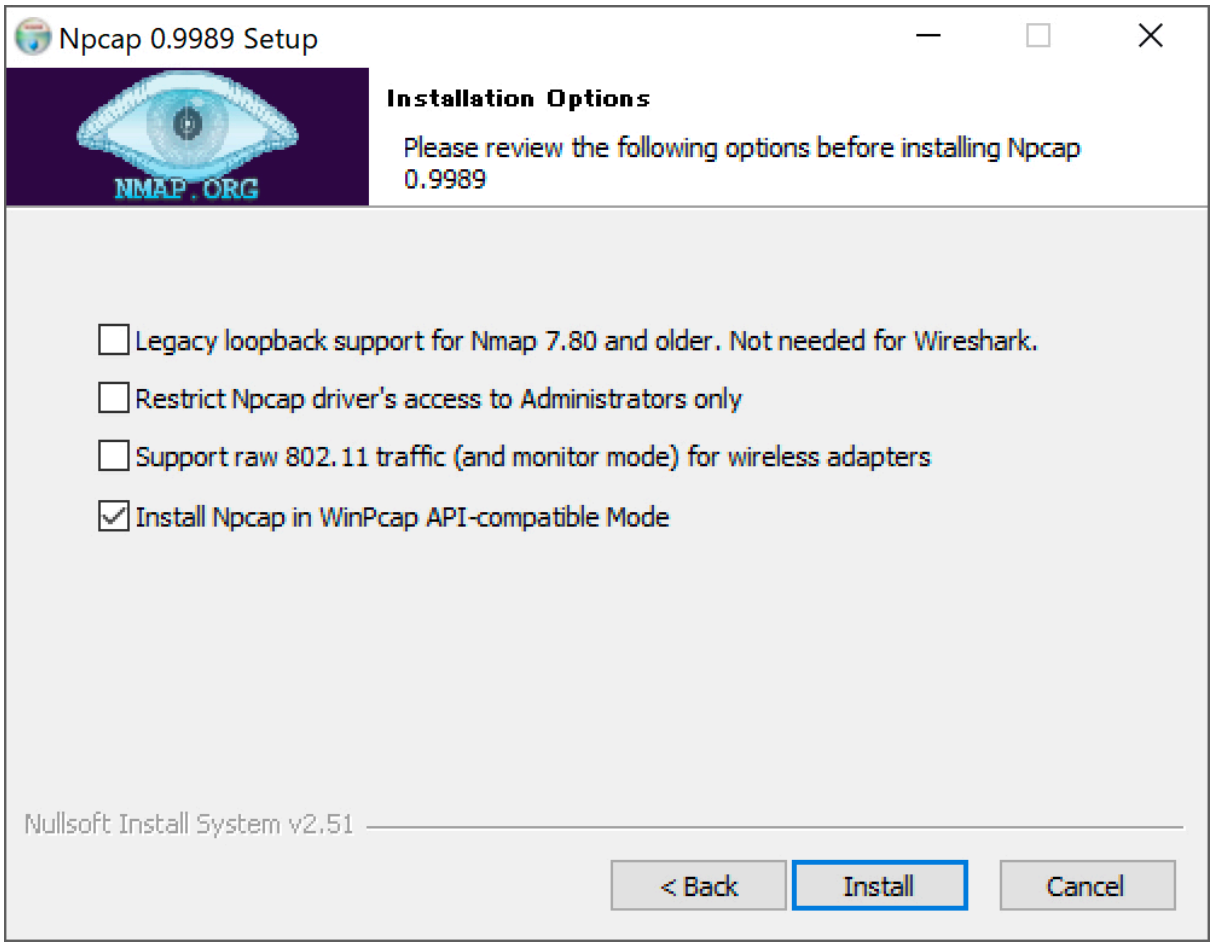

図 1　Npcap (0.9989) の WinPcap API-compatible mode

Npcap インストール後、ダウンロードしてきたインストーラ (例: `ksocket-2.0.0.0-windows-x86_64.zip`) を解凍し、中に含まれている `ksinstall.exe` をダブルクリックしてください。

上記によりインストーラーが起動するため *Ksocket* **インストーラー** に従ってインストールを進めてください。

[*5] https://nmap.org/npcap

1.1.3 Ksocket インストーラー

Ksocket インストーラーが起動すると以下のような画面が表示されます。

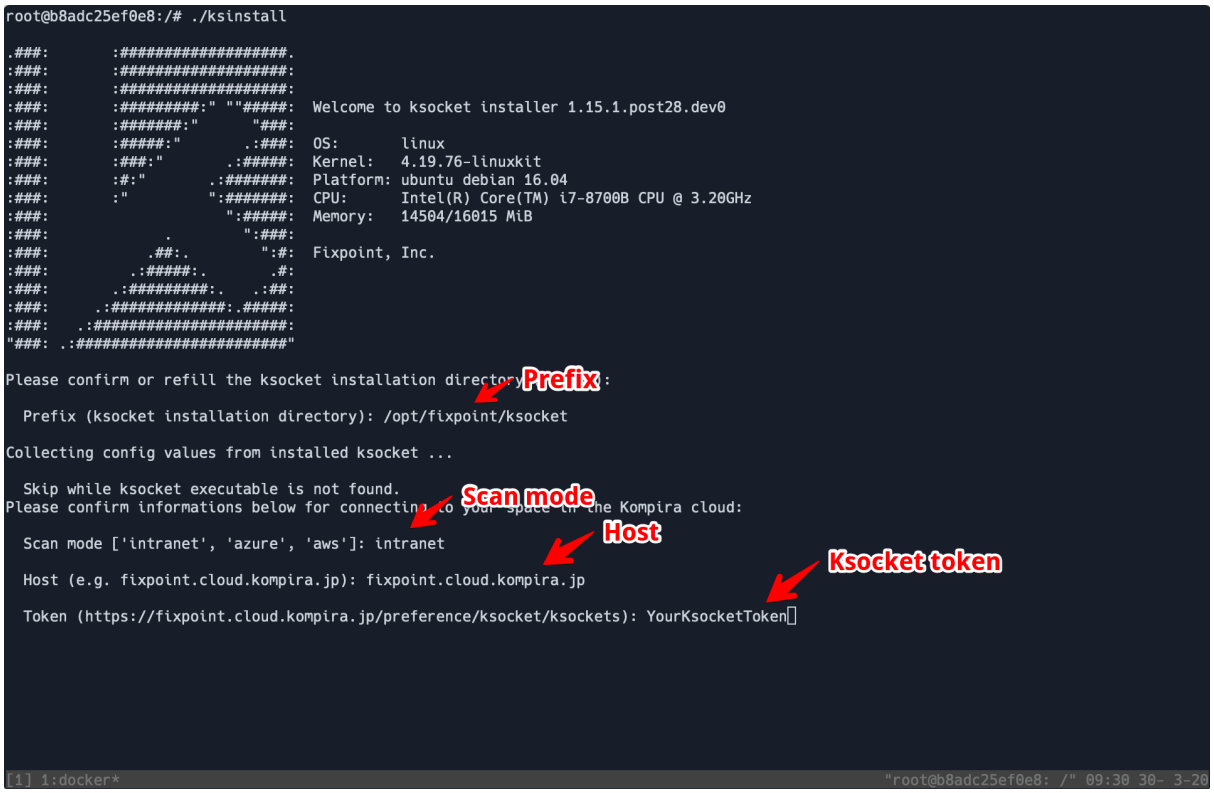

図 2　Linux にてインストーラーを起動した場合（画像は開発中のものです）

インストーラー画面にて、以下の項目を埋めてください。

Prefix　Ksocket のインストール先ディレクトリです。デフォルト値は **デフォルトのインストール先** を参照してください。

Scan mode　Ksocket のスキャン動作モードです。通常は `intranet` を入力してください。対象が Azure もしくは AWS 上の仮想ネットワークの場合は、`azure` もしくは `aws` と入力してください。**設定コマンド** にて後で有効な値を設定することもできます。

Host　接続先スペースの FQDN (Fully Qualified Domain Name) です。スペース名 + `.cloud.kompira.jp` と入力してください（例：スペース名が `fixpoint` の場合 `fixpoint.cloud.kompira.jp` ）。**設定コマンド** にて後で有効な値を設定することもできます。

Token　Ksocket 接続トークンです。Kompira cloud blog[*6] の ksocket トークンの発行[*7] を参考に有効な ksocket トークンを発行してください。**設定コマンド** にて後で有効な値を設定することもできます。

上記の入力が完了すると Ksocket のインストールが開始します。

[*6] https://blog.cloud.kompira.jp
[*7] https://blog.cloud.kompira.jp/entry/manual/ksocket-token

注釈: Windows 版 Ksocket インストーラーではクリップボードにコピーされた値を右クリックで貼り付けることができます。Linux 版では使用しているターミナルの設定に依存します。

インストール後、 Ksocket は初期化処理を行ってからインストーラにて指定されたスペースに接続します。対象スペースにアクセス後 **設定** > *Ksocket* > **対象の** *Ksocket* **スロット** のステータスが **接続済み** になっていれば、インストール完了です。

注釈: 初期化処理に必要な時間は CPU の処理能力により変わります。特に ARM 等、比較的処理能力が低い CPU では、インストール後に Kompira cloud への初回接続が行われるまで時間がかかります。

なお、インストール時に適当な `Host` や `Token` を入力した場合は、所定回数リトライした後 Ksocket サービスが停止します。正しい情報を入力して Kompira cloud に接続するには **設定コマンド** にて有効な値を設定した後 **サービス管理コマンド** の `restart` サブコマンドにて Ksocket サービスを再起動してください。

1.2 Azure へのインストール

Microsoft Azure 上に構築された AZURE VNET (Azure Virtual Network) （以下 VNET）に存在する AZURE VM (Azure Virtual Machine) （以下 VM）インスタンスをスキャンする Ksocket のインストール方法です。

スキャン対象となる VNET 上に Ksocket インストール用の新規 VM インスタンスを Ubuntu 18.04 で用意し **ローカルネットワークへのインストール** に従って Ksocket をインストールしてください。その際、スキャン動作モード選択では `azure` を指定してください。

注釈: VM インスタンスとして指定した OS 以外も利用できますが、ここでは説明の簡略化のために限定しています。

注釈: インストール時にスキャンモードの選択を間違えた場合は **設定コマンド** を参照して `azure` に再設定してください。

注釈: Ksocket は管理対象 VNET 毎にインストールする必要があります。

注釈: スキャン対象の VM 内のハードウェア構成やインストール済みパッケージの取得は、ローカルネットワークスキャンと同様に行います。そのため Ksocket に対して適切な **接続情報ファイル** を提供し SSH/WinRM/SNMP アクセスを可能にする必要があります。

注釈: スキャン対象の VM に与えられた Public IP は取得できません。これは今後のバージョンアップで改修予定です。

1.2.1 Azure リソースに対するアクセス権限の付与

スキャンモードが `azure` で動作する Ksocket は VM インスタンスに関する情報を API 経由で収集します。この際、対象の Azure のサブスクリプションに対して Reader[*8] 相当の権限を必要とします。

そのため、以下のいずれかの方法で必要な権限を付与してください。

Azure リソースのマネージド ID

Azure リソースのマネージド ID[*9] を使用し Ksocket がインストールされた VM インスタンスに対象サブスクリプションに対する権限を付与します。具体的な設定手順は下記リンクを参照してください。

- Windows VM のシステム割り当てマネージド ID を使用して Resource Manager にアクセスする[*10]
- Linux VM のシステム割り当てマネージド ID を使用して Resource Manager にアクセスする[*11]

注釈: サービスプリンシパル用のファイル（Ksocket インストール先の `./etc/ksocket/azure.toml` ）が存在する場合は削除してください。

サービスプリンシパル

サービスプリンシパルを使用し Ksocket に対象サブスクリプションに対する権限を付与します。Azure ポータルを使用してサービスプリンシパルを作成します。次に対象のサブスクリプションをスコープとするロールを、作成したサービスプリンシパルに付与します。上記の具体的な手順は下記リンクを参照してください。

- リソースにアクセスできる Azure AD アプリケーションとサービス プリンシパルをポータルで作成する[*12]
- Azure RBAC と Azure portal を使用してロールの割り当てを追加または削除する[*13]

その後 Ksocket のインストール先に `./etc/ksocket/azure.toml` を作成し、以下のように設定してください。

[*8] https://docs.microsoft.com/ja-jp/azure/role-based-access-control/built-in-roles#reader
[*9] https://docs.microsoft.com/ja-jp/azure/active-directory/managed-identities-azure-resources/overview
[*10] https://docs.microsoft.com/ja-jp/azure/active-directory/managed-identities-azure-resources/tutorial-windows-vm-access-arm
[*11] https://docs.microsoft.com/ja-jp/azure/active-directory/managed-identities-azure-resources/tutorial-linux-vm-access-arm
[*12] https://docs.microsoft.com/ja-jp/azure/active-directory/develop/howto-create-service-principal-portal
[*13] https://docs.microsoft.com/ja-jp/azure/role-based-access-control/role-assignments-portal#grant-access

```
# 作成したサービスプリンシパルの「アプリケーション ID」
clientId = "Application ID"

# 作成したサービスプリンシパルの「認証キー」
secret = "xxxxxxxxxxxxx"

# サービスプリンシパルを作成した Azure AD の「テナント ID」
tenant = "Directory ID"
```

1.3 AWS へのインストール

AWS (Amazon Web Services) 上に構築された AMAZON VPC (Amazon Virtual Private Cloud)（以下 VPC）に存在する AMAZON EC2 (Amazon Elastic Compute Cloud)（以下 EC2）インスタンスをスキャンする Ksocket のインストール方法です。

スキャン対象となる VPC に Ksocket インストール用の新規 EC2 インスタンスを Ubuntu 18.04 LTS 64 ビット (x86) で用意し **ローカルネットワークへのインストール** に従って Ksocket をインストールしてください。その際、スキャンモードの選択では `aws` を指定してください。

注釈: EC2 インスタンスとして指定した OS 以外も利用できますが、ここでは説明の簡略化のために限定しています。

注釈: インストール時にスキャンモードの選択を間違えた場合は **設定コマンド** を参照して `aws` に再設定してください。

注釈: Ksocket はスキャン対象の VPC 毎にインストールする必要があります。

注釈: スキャン対象の VPC 内のハードウェア構成やインストール済みパッケージの取得は、ローカルネットワークスキャンと同様に行います。そのため Ksocket に対して適切な **接続情報ファイル** を提供し SSH/WinRM/SNMP アクセスを可能にする必要があります。

注釈: スキャン対象の EC2 インスタンスに与えられた Public IP は取得できません。これは今後のバージョンアップで改修予定です。

1.3.1 AWS リージョンの指定

API アクセスに使用する AWS リージョンを指定します。以下のコマンドを **管理者権限** で実行して `~/.aws/config` ファイルを作成します（管理者のホームディレクトリを指すことに注意してください）：

```
% mkdir -p ~/.aws
% touch ~/.aws/config
```

その後、`~/.aws/config` を `vi` や `nano` などのテキストエディタで開き、以下のように記載／編集してください。

```
[default]
region = us-east-2
# us-east-2 の部分は利用しているリージョンコード名に置き換えてください
```

リージョン名とコードの一覧は AWS サービスエンドポイント[*14] を参照してください。

1.3.2 AWS リソースに対するアクセス権限の付与

スキャンモードが `aws` で動作する Ksocket は EC2 インスタンスに関する情報を API 経由で収集します。この際、対象の AWS リソースに対して ReadOnlyAccess[*15] 相当の権限を必要とします。

そのため、以下のいずれかの方法で必要な権限を付与してください。

IAM ロール

Amazon EC2 の IAM ロール[*16] 使用し Ksocket がインストールされた EC2 インスタンスに対象 AWS リソースに対する権限を付与します。具体的な設定手順はリンク先を参照してください。

AWS CLI プロファイル

AWS CLI プロファイルを使用し Ksocket プロセスに対象 AWS リソースに対する権限を付与します。

まず AWS CLI バージョン 2 のインストール[*17] を参考に AWS CLI を Ksocket がインストールされた EC2 インスタンスにインストールしてください。その後 **管理者権限** で AWS CLI のかんたん設定[*18] に従って設定を行ってください。これにより管理者のホームディレクトリに `~/.aws/config` および `~/.aws/credentials` が作成されます。

AWS CLI を利用せず、直接設定ファイルと認証情報ファイルを準備する場合は、以下の AWS ドキュメントを参考に該当ファイルを手動作成してください。その際管理者のホームディレクトリに対して該当ファイルを作成することに注意してください。

[*14] https://docs.aws.amazon.com/ja_jp/general/latest/gr/rande.html
[*15] https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/access_policies_managed-vs-inline.html
[*16] https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html
[*17] https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/install-cliv2.html
[*18] https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-chap-configure.html#cli-quick-configuration

- IAM ユーザーのアクセスキーの管理[*19]
- 設定ファイルと認証情報ファイルの設定[*20]

1.4 アンインストール

警告: Ksocket をアンインストールすると、設定ファイルや接続情報ファイル等も削除されます。必要に応じて、Ksocket インストール先の以下ディレクトリのバックアップを行ってください。

`./etc/ksocket` 設定ファイルや接続情報ファイル等

`./var/log/ksocket` ログファイル等

1.4.1 Linux

Ksocket はホスト Linux のパッケージを利用しないため、以下の手順でアンインストールができます。

- Ksocket サービスのアンインストール
- Ksocket インストール先ディレクトリの消去

上記手順を自動的に行う場合は、以下のコマンドを実行してください（インストール先に `/opt/fixpoint/ksocket` を利用していると想定）：

```
% /opt/fixpoint/ksocket/ksuninstall

This is a ksocket uninstaller and it will:

- Unregister ksocket service (systemd/upstart/windows service)
- Remove the files installed under '/opt/fixpoint/ksocket'

DO NOT FORGET TO BACKUP CONFIGURATIONS if you will re-install ksocket later.

  Are you sure to continue? [y/N]: y

... 略
```

[*19] https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/id_credentials_access-keys.html
[*20] https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-configure-files.html

1.4.2 Windows

Ksocket はホスト Windows のパッケージを利用しないため、以下の手順でアンインストールができます。

- Ksocket サービスのアンインストール
- Ksocket インストール先ディレクトリの消去
- Npcap のアンインストール（オプショナル）

上記手順を自動的に行う場合は **スタート** > *Fixpoint* > *Uninstall ksocket* を実行してください。なお、アンインストーラーを利用したアンインストールでは Npcap のアンインストールは行わないため、必要に応じて手動にてアンインストールしてください。

第 2 章

使用方法

注釈: この章は **設定変更や問題の調査等を目的** として Ksocket コマンドを直接実行する方法です。**ネットワークスキャン等、機能的な利用方法** に関しては Kompira cloud のマニュアルを参照してください。

- Kompira cloud マニュアル[*21]
- ローカルネットワークのスキャン[*22]
- ローカルネットワークスキャンの実行例[*23]
- ローカルネットワークスキャンの仕組み[*24]

2.1 コマンド実行方法

Ksocket はホストの環境をクリーンに保つため `$PATH` 等の環境変数を変更しません。そのため、設定／検証等で Ksocket コマンドを実行したい場合は、以下のように Ksocket シェルを立ち上げて作業する必要があります。

2.1.1 Linux

管理者権限 でターミナルを起動後、以下のコマンドで新規シェルを立ち上げてください。なお Ksocket を `/opt/fixpoint/ksocket` 以外にインストールした場合は、適時読み替えてください:

```
% PATH=/opt/fixpoint/ksocket/bin:$PATH $SHELL
```

起動したシェルにて以下のコマンドが実行できることを確認してください:

[*21] https://blog.cloud.kompira.jp/archive/category/%E3%83%9E%E3%83%8B%E3%83%A5%E3%82%A2%E3%83%AB
[*22] https://blog.cloud.kompira.jp/entry/manual/scan-local-network
[*23] https://blog.cloud.kompira.jp/entry/manual/scan-example
[*24] https://blog.cloud.kompira.jp/entry/manual/tech/scan-local-network

```
% ksocket version
2.0.0
```

2.1.2 Windows

スタートメニューの「ksocket (PowerShell)」をダブルクリックしてください。

起動した PowerShell にて以下のコマンドが実行できることを確認してください:

```
> ksocket version
2.0.0
```

2.2 設定コマンド

注釈: 以下は **コマンド実行方法** を参照し Ksocket シェルを起動した上で実行してください。

2.2.1 設定一覧の確認

設定一覧を確認するには `ksocket config` コマンドを引数なしで実行します。

以下、実行例です:

```
% ksocket config
logfile=$KSOCKET_HOME/var/log/ksocket/ksocket.log
loglevel=INFO
logformat=%(asctime)s  %(levelname)s  %(name)s:%(funcName)s:%(lineno)d  %(message)s
logfilters=ksocket
connect.token=YourKsocketToken
connect.name=ksocket
connect.protocol=wss
connect.host=fixpoint.cloud.kompira.jp
connect.port=443
connect.endpoint=%(protocol)s://%(host)s:%(port)d/api/ksocket/connect
connect.heartbeatTimeout=30.0
connect.heartbeatInterval=60.0
connect.resumeTimeout=60.0
connect.resumeRetries=120
connect.resumePlateau=10
connect.resumeThreshold=120.0
connect.requestTimeout=60.0
connect.requestRetries=10
connect.maxOperations=100
connect.maxRunningOperations=5
```

(次のページに続く)

(前のページからの続き)

```
operations.networksScanMode=intranet
directories.credentials=$KSOCKET_HOME/etc/ksocket/credentials
```

上記にて = より左側が「設定名」で右側が「設定値」です。

2.2.2 設定の確認

任意の設定値を確認するには `ksocket config` コマンドを、確認したい設定名と共に呼び出します。

例えば `connect.host` の値を確認したい場合は以下のようにします:

```
% ksocket config connect.host
fixpoint.cloud.kompira.jp
```

2.2.3 設定の変更

注釈: Ksocket は起動時に設定ファイルを読むため、設定を適用するには Ksocket を再起動してください。

設定の変更を行うには `ksocket config` コマンドを変更対象の設定名および設定値と共に呼び出します。

例えば `connect.host` の値を `myspace.cloud.kompira.jp` に変更したい場合は以下のようにします:

```
% ksocket config connect.host myspace.cloud.kompira.jp
```

値にスペースや特殊文字等を含む場合は各シェルのエスケープルールに従って、適切にエスケープしてください。

例: Linux (Bash):

```
% ksocket config connect.token "e.g. \$, \", and \\"
```

例: Windows (PowerShell):

```
> ksocket config connect.token "e.g. ^$, \^", and ^^"
```

2.3 自己診断コマンド

注釈: 以下は **コマンド実行方法** を参照し Ksocket シェルを起動した上で実行してください。

問題調査等で Ksocket の動作確認を行いたい場合は `ksocket dignostics` コマンドを実行してください。下記で説明する `system`, `config` および `connect` コマンドを順次実行します。なお `ksocket dignostics` コマンドは Ksocket スロットの一時的な占有を防ぐため `authenticate` を実行しません。認証まで含めて診断する場合は、続けて `ksocket dignostics authenticate` コマンドを実行してください。

2.3.1 システム診断 (system)

システム的な不備がないか診断するには `ksocket dignostics system` を呼び出してください。以下のようにすべて `OK` であれば、システム的な問題はありません:

```
% ksocket dignostics system
Python:  3.7.6 (default, Mar  4 2020, 12:52:25)  [GCC 4.4.7 20120313 (Red Hat 4.4.
↪7-23)]
Ksocket: 2.0.0

KSOCKET_HOME:    /opt/fixpoint/ksocket
KSOCKET_CONFIG:  /opt/fixpoint/ksocket/etc/ksocket/ksocket.toml

$PYTHONHOME:
$PYTHONPATH:
$PYTHONOPTIMIZE: 2
$PYTHONENCODING:
$PYTHONUTF8:
$KSOCKET_HOME:   /opt/fixpoint/ksocket
$KSOCKET_CONFIG:

ssl ................... OK
aiofiles .............. OK
aiohttp ............... OK
asyncssh .............. OK
dpkt .................. OK
netifaces ............. OK
Cryptodome ............ OK
pcap .................. OK
pysnmp ................ OK
pytz .................. OK
winrm ................. OK
yaml .................. OK
requests_credssp ...... OK
ntlm_auth ............. OK
jsonref ............... OK
stringcase ............ OK
```

(次のページに続く)

(前のページからの続き)

```
websockets ............ OK
websockets.speedups ... OK
```

2.3.2 設定ファイル診断 (config)

設定ファイルに不備がないか診断するには `ksocket dignostics config` を呼び出してください。以下のようにすべて `OK` であれば、設定ファイルに問題はありません:

```
% ksocket dignostics config
Python:  3.7.6 (default, Mar  4 2020, 12:52:25)  [GCC 4.4.7 20120313 (Red Hat 4.4.
↪7-23)]
Ksocket: 2.0.0

KSOCKET_HOME:    /opt/fixpoint/ksocket
KSOCKET_CONFIG:  /opt/fixpoint/ksocket/etc/ksocket/ksocket.toml

$PYTHONHOME:
$PYTHONPATH:
$PYTHONOPTIMIZE: 2
$PYTHONENCODING:
$PYTHONUTF8:
$KSOCKET_HOME:   /opt/fixpoint/ksocket
$KSOCKET_CONFIG:

Read KSOCKET_CONFIG as bytes .......... OK
Read KSOCKET_CONFIG as UTF-8 .......... OK
Read KSOCKET_CONFIG as TOML/YAML ...... OK
Read KSOCKET_CONFIG as config file .... OK
```

2.3.3 疎通状況診断 (connect)

DNS 設定等 Kompira cloud との疎通に不備がないか診断するには `ksocket dignostics connect` を呼び出してください。以下のようにすべて `OK` であれば、疎通状況に問題はありません:

```
% ksocket dignostics connect
Python:  3.7.6 (default, Mar  4 2020, 12:52:25)  [GCC 4.4.7 20120313 (Red Hat 4.4.
↪7-23)]
Ksocket: 2.0.0

KSOCKET_HOME:    /opt/fixpoint/ksocket
KSOCKET_CONFIG:  /opt/fixpoint/ksocket/etc/ksocket/ksocket.toml

$PYTHONHOME:
$PYTHONPATH:
$PYTHONOPTIMIZE:
$PYTHONENCODING:
$PYTHONUTF8:
```

(次のページに続く)

(前のページからの続き)

```
$KSOCKET_HOME:   /opt/fixpoint/ksocket
$KSOCKET_CONFIG:

Resolve DNS forward lookup (cloud.kompira.jp) ..... OK [40.81.216.149]
Establish TCP connection (cloud.kompira.jp:80) .... OK
Establish TCP connection (cloud.kompira.jp:443) ... OK

KSOCKET_INSECURE: False

Host:     fixpoint.cloud.kompira.jp
Port:     443
Protocol: wss
Endpoint: wss://fixpoint.cloud.kompira.jp:443/api/ksocket/connect

Resolve DNS forward lookup ... OK [40.81.216.149]
Establish TCP connection ..... OK
Establish websocket .......... OK
```

2.3.4 認証情報診断 (authenticate)

重要: 認証処理を行うため、すでに同じ Ksocket スロットに対して別の Ksocket が接続している場合は失敗します。認証情報診断を行う際は新規 Ksocket スロットを作成して、そのスロットに対して接続を行ってください。

Ksocket token の誤り等、認証情報に不備がないか診断するには `ksocket dignostics authenticate` を呼び出してください。以下のようにすべて `OK` であれば、認証情報に問題はありません:

```
% ksocket dignostics authenticate
Python:  3.7.6 (default, Mar  4 2020, 12:52:25)  [GCC 4.4.7 20120313 (Red Hat 4.4.
↪7-23)]
Ksocket: 2.0.0

KSOCKET_HOME:    /opt/fixpoint/ksocket
KSOCKET_CONFIG:  /opt/fixpoint/ksocket/etc/ksocket/ksocket.toml

$PYTHONHOME:
$PYTHONPATH:
$PYTHONOPTIMIZE:
$PYTHONENCODING:
$PYTHONUTF8:
$KSOCKET_HOME:   /opt/fixpoint/ksocket
$KSOCKET_CONFIG:

KSOCKET_INSECURE: False

Host:     fixpoint.cloud.kompira.jp
```

(次のページに続く)

(前のページからの続き)

```
Port:     443
Protocol: wss
Endpoint: wss://fixpoint.cloud.kompira.jp:443/api/ksocket/connect

Complete authentication ...... OK
```

2.4 サービス管理コマンド

注釈: 以下は **コマンド実行方法** を参照し Ksocket シェルを起動した上で実行してください。

Ksocket が利用しているデーモン／サービスを管理するためのコマンドです。各プラットフォームに合わせて systemd[*25], Upstart[*26] もしくは NSSM[*27] をバックエンドとして使用します。以下、Ksocket デーモン／サービスを Ksocket サービスと呼称します。

2.4.1 インストール (`install`)

Ksocket サービスをインストールします。成功時は以下のように表示されます（プラットフォームによって若干表示内容に差があります）：

```
% ksocket service install
Check if ksocket runner exists .......... OK
Create a service file ................... OK [/opt/fixpoint/ksocket/usr/lib/
↪systemd/system/ksocket.service]
Install the service ..................... OK
Get status of the service ............... OK
Restart the service ..................... OK
```

Linux では systemd[*28] もしくは Upstart[*29] のうち、検出できた方に対して `ksocket` という名前のデーモンを登録します。各設定ファイルのインストール先は以下を参照してください。

systemd `/usr/lib/systemd/system/` (CentOS/RedHat) `/lib/systemd/system/` (Ubuntu/Debian) `/usr/local/lib/systemd/system/` (FreeDesktop)

Upstart `/etc/init`

Windows では Ksocket にバンドルされている NSSM[*30] を利用して `Fixpoint ksocket service` という名前のサービスを登録します。確認する場合は **スタート** > *Windows* **管理ツール** > **サービス** から確認してください。

[*25] https://www.freedesktop.org/wiki/Software/systemd/
[*26] http://upstart.ubuntu.com/
[*27] https://nssm.cc/
[*28] https://www.freedesktop.org/wiki/Software/systemd/
[*29] http://upstart.ubuntu.com/
[*30] https://nssm.cc/

注釈: Windows にて CPU 使用率が異常に高い場合（シングルコアに対して 4 スレッドで 100% の使用率等）に Ksocket サービスのインストールに失敗することがあります。その場合は CPU 使用率が下がってから再度サービスインストールを実行してください。

2.4.2 アンインストール (uninstall)

Ksocket サービスをアンインストールします。成功時は以下のように表示されます（プラットフォームによって若干表示内容に差があります）：

```
% ksocket service uninstall
Stop the service ......................... OK
Uninstall the service .................... OK

Remove a service file .................... OK [/opt/fixpoint/ksocket/usr/lib/
↪systemd/system/ksocket.service]
```

2.4.3 サービスの状態確認 (status)

Ksocket サービスの状態を確認します。以下のいずれかの状態が表示されます。

missing　Ksocket サービスはインストールされていません。

running　Ksocket サービスは作動しています。

stopped　Ksocket サービスは停止しています。

processing　Ksocket サービスは状態変化中です。

以下、作動中に実行した例:

```
% ksocket service status
running
```

2.4.4 サービスの開始 (start)

Ksocket サービスを開始します。成功時は以下のように表示されます（プラットフォームによって若干表示内容に差があります）：

```
% ksocket service start
Get status of the service ............... OK
Start the service ....................... OK
```

既に Ksocket サービスが開始されていた場合は `Skip` と表示され、成功扱いになります:

```
% ksocket service start
Get status of the service ..... OK
Start the service ..... Skip
```

2.4.5 サービスの停止 (stop)

Ksocket サービスを停止します。成功時は以下のように表示されます（プラットフォームによって若干表示内容に差があります）：

```
% ksocket service stop
Get status of the service ..... OK
Stop the service ..... OK
```

既に Ksocket サービスが停止されていた場合は `Skip` と表示され、成功扱いになります:

```
% ksocket service stop
Get status of the service ..... OK
Stop the service ..... Skip
```

2.4.6 サービスの再起動 (restart)

Ksocket サービスを再起動します。成功時は以下のように表示されます（プラットフォームによって若干表示内容に差があります）：

```
% ksocket service restart
Get status of the service ..... OK
Restart the service ..... OK
```

既に Ksocket サービスが停止されていた場合もリスタート扱いとなります。

第 3 章

設定ファイル

Ksocket は各種設定ファイルを TOML[*31] 0.4.0 形式で記載します。

注釈: 後方互換性のため YAML[*32] 1.1 形式もサポートしていますが、非推奨機能のため YAML/TOML コンバーター（例：https://fixpoint.github.io/yaml-toml-converter/）等を用いて TOML 形式に変換することをお勧めします。

3.1 Ksocket 設定

Ksocket は起動時に Ksocket インストール先の `./etc/ksocket/ksocket.toml` に記載された設定を読み込みます。設定の確認／変更には **設定コマンド** を使用してください。

> **警告:** 下記にて明言されていない設定項目も存在しますが、内部的な項目のため変更しないでください。

3.1.1 `logfile`

ログファイルの書き込み先ファイルを指定します。値として - を指定した場合、ファイルではなく標準出力に結果を書き出します。デフォルト値は `$KSOCKET_HOME/var/log/ksocket/ksocket.log` です。

`ksocket` コマンドとして実行する場合は `--logfile={VALUE}` というコマンド引数で上書きできます。

以下の特殊変数が利用可能です

`$KSOCKET_HOME` Ksocket のインストール先ディレクトリパス

[*31] https://github.com/toml-lang/toml
[*32] https://yaml.org/

3.1.2 loglevel

ログの出力レベルを DEBUG, INFO, WARNING, ERROR, CRITICAL から指定します。検証時は DEBUG か INFO 、運用時は WARNING 以上に設定することをお勧めします。デフォルト値は INFO です。

ksocket コマンドとして実行する場合は --loglevel={VALUE} というコマンド引数で上書きできます。

3.1.3 connect.token

Kompira cloud との接続時に利用する Ksocket トークンを指定します。

Ksocket トークンの発行は Ksocket トークンの発行[*33] を参照してください。

3.1.4 connect.host

接続先の Kompira cloud スペースを FQDN で指定します(例:スペース名が fixpoint の場合は fixpoint.cloud.kompira.jp)。

3.1.5 connect.port

Kompira cloud との接続時に利用するポート番号を指定します。特別な事情がない限り 443 を利用してください。デフォルト値は 443 です。

3.1.6 connect.protocol

Kompira cloud との接続時に利用するプロトコルを wss, ws から指定します。特別な事情がない限り wss を利用してください。デフォルト値は wss です。

3.1.7 operations.networksScanMode

ネットワークスキャンの際に利用するモードを intranet, azure, aws から指定します。スキャン対象のネットワーク種類によって切り替えてください。

intranet　社内ネットワーク等のローカルネットワークをスキャンする場合

azure　Microsoft Azure (Azure) 上の Azure Virtual Machine (Azure VM) インスタンスを API 経由でスキャンする場合

aws　AWS 上の Amazon EC2 インスタンスを API 経由でスキャンする場合

[*33] https://blog.cloud.kompira.jp/entry/manual/ksocket-token

3.1.8 directories.credentials

クレデンシャルファイルの保存先ディレクトリを指定します。デフォルト値は `$KSOCKET_HOME/etc/ksocket/credentials` です。

以下の特殊変数が利用可能です

`$KSOCKET_HOME` Ksocket のインストール先ディレクトリパス

3.2 接続情報

Ksocket はリモートホストに対する接続試行時に Ksocket インストール先の `./etc/ksocket/credentials` から対象ホストにマッチする接続情報ファイルを読み込みます。

詳細情報取得タイミングに関しては ローカルネットワークスキャンの仕組み[*34] を参照してください。

接続情報ファイルの作成には Kompira cloud ログイン後の **設定** > **ツール** > **認証情報生成** を利用してください。

3.2.1 SSH (Linux に対しての接続)

SSH (Secure Shell) を使用したリモートホストへのアクセスに関する接続情報を指定します。この接続情報ファイルは Ksocket インストール先の `./etc/ksocket/credentials/ssh` 以下に保存してください。

このファイルは以下のようなフォーマットで作成します。

```
# IP addresses/Networks which use this credential file
# この認証情報を利用して接続する IP アドレスもしくはネットワークの一覧
# 以下例では 10.10.0.0/24 のホスト及び 10.20.0.1, 10.20.0.3 が指定されている
includes = ["10.10.0.0/24", "10.20.0.1", "10.20.0.3"]

# 除外対象とする IP アドレスもしくはネットワークの一覧
# includes よりも優先されるため includes で大きな範囲を指定した後に
# excludes で除外設定を行うことが可能
#excludes = ["10.10.0.1", "10.10.0.3"]

# 接続に使用するポート番号。省略時は 22
#port = 10022

# 接続のタイムアウト時間
#timeout = 5.0

# 踏み台とするホストの IP アドレスリスト
# 以下例のように指定すると、以下の経路で SSH 接続を行う
#
#   ksocket --> 10.10.0.1 --> 10.10.0.3 --> (target IP)
```

(次のページに続く)

[*34] https://blog.cloud.kompira.jp/entry/manual/tech/scan-local-network

(前のページからの続き)

```
#
#sshTunnels = ["10.10.0.1", "10.10.0.3"]

# 接続に使用するアカウント情報
[account]
# ユーザ名
username = "john"

# パスワード
password = "passw0rd"

# 接続に使用する鍵ファイル候補一覧
# 鍵ファイルを使用しない場合は以降すべてを削除もしくはコメントアウトする
[[account.clientKeys]]
filename = "../../id_rsa.common"
passphrase = "Common password"

[[account.clientKeys]]
filename = "/root/.ssh/id_rsa"
passphrase = "Admin password"
```

上記において、 `account.clientKeys.filename` には絶対パスおよび相対パスが指定可能です。

3.2.2 WinRM (Windows に対しての接続)

WINRM (Windows Remote Management) を使用したリモートホストへのアクセスに関する接続情報を指定します。この接続情報ファイルは Ksocket インストール先の `./etc/ksocket/credentials/winrm` 以下に保存してください。

注釈: WinRM で詳細情報を取得する場合は、スキャン対象ホストにて *WinRM* **接続の有効化** 等を参考に WinRM の有効化を行ってください。

このファイルは以下のようなフォーマットで作成します。

```
# この認証情報を利用して接続する IP アドレスもしくはネットワークの一覧
# 以下例では 10.10.0.0/24 のホスト及び 10.20.0.1, 10.20.0.3 が指定されている
includes = ["10.10.0.0/24", "10.20.0.1", "10.20.0.3"]

# 除外対象とする IP アドレスもしくはネットワークの一覧
# includes よりも優先されるため includes で大きな範囲を指定した後に
# excludes で除外設定を行うことが可能
#excludes = ["10.10.0.1", "10.10.0.3"]

# 接続方法
# basic: ベーシック認証 (デフォルト)
# ntlm: NT LAN Manager (NTLM) 認証
# credssp: Credential Security Support Provider (CredSSP) 認証
```

(次のページに続く)

(前のページからの続き)

```
#authMethod = "basic"

# メッセージ暗号化設定
# auto: 暗号化を行う (デフォルト)
# always: 常に暗号化を行う
# never: 暗号化を行わない
#authMessageEncryption = "auto"

# authMethod: credssp 指定時のみ有効。
# true を指定すると、TLSv1.2 による通信を無効化する。
# 主に Windows Server 2008 に接続する際に使用
#authCredSSPDisableTLSv1.2 = false

# 接続に使用するポート番号。省略時は 5985
#port = 5985

# 接続のタイムアウト時間
#timeout = 5.0

# SSH 踏み台 IP アドレスリスト
# 以下例のように指定すると、以下の経路で SSH/WinRM 接続を行う
#
#   ksocket --> 10.10.0.1 --> 10.10.0.3 --> (target IP)
#
# 終端の IP アドレスへのアクセスのみが WinRM 接続となり、
# そこまでの各経路では SSH 接続を行う。
#sshTunnels = ["10.10.0.1", "10.10.0.3"]

# 接続に使用するアカウント情報
[account]
# アカウント名
# ドメインアカウントを使用する場合、
# 'MYDOMAIN\\USER01' (NTLM 形式)
# 'user01@MYDOMAIN' (UPN 形式)
# のように指定する。
username = "john"

# パスワード
password = "passw0rd"
```

注釈: ドメインアカウントを使用する場合、Basic 認証によるアクセスは行うことができません。この場合は NTLM 認証でアクセスするように接続情報を設定してください。`authMethod` パラメータを `ntlm` と指定することで、NTLM 認証でのアクセスとなります。

3.2.3 SNMP (ネットワーク機器に対しての接続)

SNMP (Simple Network Management Protocol) を使用したリモートホストへのアクセスに関する接続情報を指定します。この接続情報ファイルは Ksocket インストール先の `./etc/ksocket/credentials/snmp` 以下に保存してください。

このファイルは以下のようなフォーマットで作成します。

```
# この認証情報を利用して接続する IP アドレスもしくはネットワークの一覧
# 以下例では 10.10.0.0/24 のホスト及び 10.20.0.1, 10.20.0.3 が指定されている
includes = ["10.10.0.0/24", "10.20.0.1", "10.20.0.3"]

# 除外対象とする IP アドレスもしくはネットワークの一覧
# includes よりも優先されるため includes で大きな範囲を指定した後に
# excludes で除外設定を行うことが可能
#excludes = ["10.10.0.1", "10.10.0.3"]

# 接続に使用するポート番号。省略時は 161
#port = 161

# 接続のタイムアウト時間
#timeout = 5.0

# 接続リトライ回数
#retries = 0

# 接続リトライ間隔 (秒)
#retryInterval = 1.0

[authData]
# 接続に使用するコミュニティ名 (SNMP v2c の場合のみ指定)
community = "public"

# 以下、SNMP v3 の場合のみ指定
# ユーザ名
username = "your-username"

# 認証方式
# usmNoAuthProtocol             (認証なし、default)
# usmHMACMD5AuthProtocol        (MD5(HMAC-MD5-96) に対応)
# usmHMACSHAAuthProtocol        (SHA(HMAC-SHA-96) に対応)
# usmHMAC128SHA224AuthProtocol
# usmHMAC192SHA256AuthProtocol
# usmHMAC256SHA384AuthProtocol
# usmHMAC384SHA512AuthProtocol
authProtocol = "usmHMACMD5AuthProtocol"

# 認証パスワード
#authKey = "your-password"

# 暗号化方式
# usmNoPrivProtocol       (暗号化なし、default)
```

(次のページに続く)

(前のページからの続き)

```
# usmDESPrivProtocol      (DES(CBC-DES))
# usm3DESEDEPrivProtocol (3DES-EDE)
# usmAesCfb128Protocol    (AES(CFB128-AES-128))
# usmAesCfb192Protocol    (AES(CFB128-AES-192))
# usmAesCfb256Protocol    (AES(CFB128-AES-256))
privProtocol = "usmAesCfb128Protocol"

# 暗号化パスワード
privKey = "priv-your-password"
```

3.2.4 接続情報ファイルの検索順

リモートホストに対する接続を行うオペレーションが実行されると、接続対象の IP アドレスをもとに適切な接続情報ファイルが検索されます。この検索は、以下のルールに基づき行われます。

1. Ksocket インストール先の `./etc/ksocket/credentials/<`**接続方式名**`>` 内のファイル・フォルダを辞書順に検索
2. ファイル名・フォルダ名が _ から始まる場合は除外
3. ディレクトリの場合
    1. ディレクトリ内のファイルに対して同様の検索（幅優先）
4. ファイルの場合は以下の適合テストに移行
    1. ファイル名が `.toml` または `.yml` で終わっていない場合は除外
    2. `includes` の値を読み取り、対象の IP アドレスが含まれていない場合は除外
    3. `excludes` の値を読み取り、対象の IP アドレスが含まれている場合は除外
    4. 上記にて除外されなかった場合は適合と判定

上記の検索にて適合した接続情報ファイルは `INFO` レベルでログに記載されます。また、適合する接続情報ファイルが見つからなかった場合は `WARNING` レベルでログに記載され、リモートへの接続を試みる前にオペレーションが終了します。

3.2.5 接続情報ファイルの検索例

以下のような構成で接続情報ファイルが保存されていると仮定します。なお `$KSOCKET_HOME` は Ksocket インストール先を示すものとします。

```
+- $KSOCKET_HOME/etc/ksocket/credentials/ssh
    +- 00_common.toml
    +- 01_specific/
    |   +- 00_common.toml
    |   +- 02_final.toml
```

(次のページに続く)

(前のページからの続き)

```
|   +- _special.toml
+- 02_final.toml
+- 99-example.toml.skeleton
+- _projectA/
|   +- 00_common.toml
|   +- 02_final.toml
|   +- _special.toml
+- _projectB/
    +- 00_common.toml
    +- 02_final.toml
    +- _special.toml
```

この場合

1. `./00_common.toml`
2. `./01_specific/00_common.toml`
3. `./01_specific/02_final.toml`
4. `./02_final.toml`

の順で適合テストが行われ、適合した順にリモートホストへの接続試行を行います。

第 4 章

トラブルシュート

4.1 ksocket コマンドが見つからない

ksocket コマンドを実行した場合以下のように ksocket コマンドが見つからないというメッセージが表示されることがあります。

bash	-bash ksocket: command not found
PowerShell	ksocket : 用語 'ksocket' は、コマンドレット、関数、スクリプト ファイル、または操作可能なプログラムの名前として認識されません。名前が正しく記述されていることを確認し、パスが含まれている場合はそのパスが正しいことを確認してから、再試行してください。
cmd	'ksocket' は、内部コマンドまたは外部コマンド、操作可能なプログラムまたはバッチ ファイルとして認識されていません。

これは Ksocket がホストの環境をクリーンに保つため $PATH 等の環境変数を変更しないことに起因します。詳細は **コマンド実行方法** を参照してください。

4.2 自己診断コマンドによる診断

Ksocket にまつわる問題を調査する場合は **自己診断コマンド** を参照し自己診断コマンドを実行してください。

4.2.1 system 診断が失敗する

システム的な不備があります。自己診断コマンドの出力結果と共にサポートまでお問い合わせください。

4.2.2 config 診断が失敗する

設定ファイルに不備があります。表示されたメッセージ等を参考に設定ファイルを修正してください。

4.2.3 connect 診断が失敗する

疎通状況に不備があります。DNS やファイアーウォール等のネットワーク設定を見直してください。なお Ksocket 単体では HTTP Proxy 配下で利用できません。HTTP Proxy 下で利用する場合は プロキシ経由で ksocket を使用する[*35] を参考に ksbridge を利用してください。

4.2.4 authenticate 診断が失敗する

Ksocket の認証処理に不備があります。

- Ksocket トークンが失効していないか確認してください
- Ksocket スロットに対して既に接続が存在しないか確認してください（Ksocket スロットに対して同時に接続できる Ksocket は一つのみです）

4.3 デバッグログによる調査

ログレベルを変更するために **サービス管理コマンド** の `stop` サブコマンドを利用して Ksocket サービスを停止します:

```
% ksocket service stop
Get status of the service ..... OK
Stop the service ..... OK
```

停止後に **設定コマンド** を利用して `loglevel` を `DEBUG` に変更してください:

```
% ksocket config loglevel DEBUG
```

その後、現在のログファイルをリネームしてから Ksocket サービスを起動します:

```
% cd /opt/fixpoint/ksocket
% mv var/log/ksocket/ksocket.log var/log/ksocket/ksocket.prev.log
% ksocket service start
Get status of the service ..... OK
Start the service ..... OK
```

[*35] https://blog.cloud.kompira.jp/entry/manual/ksocket-with-proxy

これにより Ksocket インストール先の `./etc/ksocket/ksocket.log` に詳細なデバッグログが記録されるようになったため、スキャン等を実行してログを収集してください。

4.4 サポートへ問い合わせ

Fixpoint コミュニティサイト[*36] の Sonar 関連トピックもしくは support@kompira.jp にお問い合わせください。

なお、問い合わせの際に以下の情報を追加していただけると、スムーズな問題解決につながります。

- 利用しているスペース名
- 自己診断コマンドの結果
- DEBUG レベルで出力した Ksocket ログ
- スキャンを行った大まかな日時

[*36] https://kompira.zendesk.com/hc/ja/community/topics

第 5 章

変更履歴

5.1 Ksocket v2.0.0

リリース日 2020/04/01

主に近日リリース予定の Ksocket リモートアップデート機能対応と ARMv7 のサポートを含んだリリースです。多数の非互換変更を含むため **Ksocket v1 を利用していた方はマイグレーションガイドを参照してください**。

Ksocket v1 **から** *v2* **への移行**

5.1.1 新機能

- CPU アーキテクチャとして ARMv7 Little Endian をサポート
- 自己診断コマンドの追加
- デーモン／サービス管理コマンドの追加
- 上書きインストールに対応
- Windows にて RSA 鍵の読み取り権限を ACL を用いて ksocket 実行ユーザーおよび Administrator に限定する機能を追加
- 内部的に利用するため `KSOCKET_CONFIG` 環境変数による読み取り設定ファイルの切り替え機能追加
- 内部的に利用している `insecure` オプションの廃止および `KSOCKET_INSECURE` 環境変数の追加
- 内部的なオペレーション同時実行数制限 (`maxRunningOperations`) の導入

5.1.2 変更点

- セマンティックバージョニングの採用
- 設定ファイルの読み込みに失敗した場合にプロセスが終了するように変更 - **非互換**
- Azure/AWS の自動判定機能の廃止およびスキャンモードの追加 - **非互換**
- 内部的な設定項目名を snake_case から camelCase に変更
- 内部的な再配置可能 Python を Conda から独自形式に変更
- 内部的な Windows サービスの登録方式を pywin32 方式から NSSM 方式に変更

5.1.3 削除予定の機能

- YAML 形式の設定ファイル

5.1.4 削除された機能

- 内部的な設定項目を一部廃止 - **非互換**
- 内部的なコマンドの一部引数を廃止 - **非互換**
- 内部的なオペレーションプラグイン機構廃止 - **非互換**
- 内部的なコマンドプラグイン機構廃止 - **非互換**
- 内部的な一部オペレーションの廃止 - **非互換**

5.1.5 バグ修正

- 一部のアドレス存在確認プロトコルにて OS のルーティングテーブルを考慮しない問題を修正
- ネットワーク遅延等によりオペレーションが重複発行された場合にスキャンが不安定になる問題を修正
- 大量のアドレスをスキャンした際にスキャンが不安定になる問題を修正

5.1.6 脆弱性対応

なし

第 6 章

付録

6.1 デフォルトのインストール先

Windows `C:\ProgramData\Fixpoint\ksocket`

Linux `/opt/fixpoint/ksocket`

6.2 WinRM 接続の有効化

自動で WinRM を有効化する場合は以下を参照してください。

- WinRM 接続を簡単に有効化するスクリプト[*37]
- Ksocket 接続受け入れの一括設定マニュアル[*38]

手動にて有効化する場合は WinRM 接続対象の Windows にて PowerShell を管理者権限で開き、以下を実行してください:

```
# ExecutionPolicy が Restricted だった場合、RemoteSigned に変更する
> Get-ExecutionPolicy
Restricted
> Set-ExecutionPolicy RemoteSigned
> Get-ExecutionPolicy
RemoteSigned

# WinRM サービスを実行できるようにする
> winrm qc

# Basic 認証で接続する場合は、Basic 認証での接続を許可する
> winrm set winrm/config/service/auth '@{Basic="true"}'
> winrm set winrm/config/service '@{AllowUnencrypted="true"}'

# ユーザに対して読み取り権限を付与する
```

(次のページに続く)

[*37] https://blog.cloud.kompira.jp/entry/utility/winrm-config-script
[*38] https://blog.cloud.kompira.jp/entry/manual/winrm-bulk-setting-manual

(前のページからの続き)

```
# 以下コマンド実行によって表示されたウィンドウで、
# 該当するユーザに読み取り権限と実行権限を許可して適用
> winrm configSDDL default

# WMI リソースのアクセス権限設定
# 以下コマンド実行によって表示されたウィンドウで、
# [操作]>[プロパティ]>[セキュリティ] を選択
# - Root\CIMV2 から [セキュリティ] を選択し、
#   該当するユーザにメソッドの実行とリモートの有効化を許可して適用
# - Root\StandardCimv2 から [セキュリティ] を選択し、
#   該当するユーザにメソッドの実行とリモートの有効化を許可して適用
> wmimgmt.msc
```

6.3 用語集

Ksocket トークン　Ksocket が Kompira cloud に接続する際に利用する認証文字列。API トークンとは異なるものなので注意。

設定ファイル　Ksocket が Kompira cloud に接続するための情報や、ログの出力方法などが記載されたファイル。記載方法等、詳細は *Ksocket* **設定** を参照。

接続情報ファイル　Ksocket がスキャン対象機器にログインするための情報が記載されたファイル。記載方法等、詳細は **接続情報** を参照。

6.4 Ksocket v1 から v2 への移行

既に Ksocket v1 がインストールされている場合は、まず **設定ファイルおよび設定情報ファイルをバックアップ** してください。その後、対応する Ksocket ドキュメントに従って既存の Ksocket を **アンインストール** してから Ksocket v2 を **新規インストール** してください。

6.4.1 設定ファイルの移行

Ksocket v2 では設定ファイルに対して以下の変更点があります。全て内部的な設定値なので、利用していない場合は無視してください。

Ksocket v1	Ksocket v2
connect.insecure	KSOCKET_INSECURE 環境変数を代わりに利用してください
connect.heartbeat_timeout	connect.heartbeatTimeout
connect.heartbeat_interval	connect.heartbeatInterval
connect.resume_timeout	connect.resumeTimeout
connect.resume_retries	connect.resumeRetries
connect.resume_plateau	connect.resumePlateau
connect.resume_threshold	connect.resumeThreshold
connect.request_timeout	connect.requestTimeout
connect.request_retries	connect.requestRetries
connect.max_operations	connect.maxOperations

6.4.2 接続情報ファイルを暗号化している場合

Ksocket v2 は **新規インストール** が必要なため、接続情報ファイルを ksocket encrypt を利用して暗号化している場合は ksocket decrypt を利用して復号化してください。暗号化／復号化の詳細は https://blog.cloud.kompira.jp/entry/2019/08/24/133651 を参照してください。

6.4.3 systemd を利用している場合

systemd を利用していて ksocket.service ファイルを **カスタマイズしている場合** は KillMode=process を [Service] セクションに加えてください。以下 ksocket.service の例です:

```
[Unit]
Description=A RPC like client service for Fixpoint, Inc. Kompira cloud
After=network.target

[Service]
Type=simple
KillMode=process
WorkingDirectory=/opt/fixpoint/ksocket
ExecStart=/opt/fixpoint/ksocket/bin/ksocket connect
StartLimitInterval=10
StartLimitBurst=5

[Install]
WantedBy=multi-user.target
```

6.4.4 Azure/AWS で利用している場合

新規インストール時の `Scan mode` 選択で `azure` もしくは `aws` を入力してください。

索引