ksocket Documentation

リリース ***20190521***

Fixpoint, Inc.

2019 年 **05** 月 **21** 日

目次

ksocket は 株式会社フィックスポイント 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** と連携して動作することが前提となっており、 **Kompira cloud** からの実行命令を受けて **ksocket** はネットワーク内の情報を収集して **Kompira cloud** に送信します。

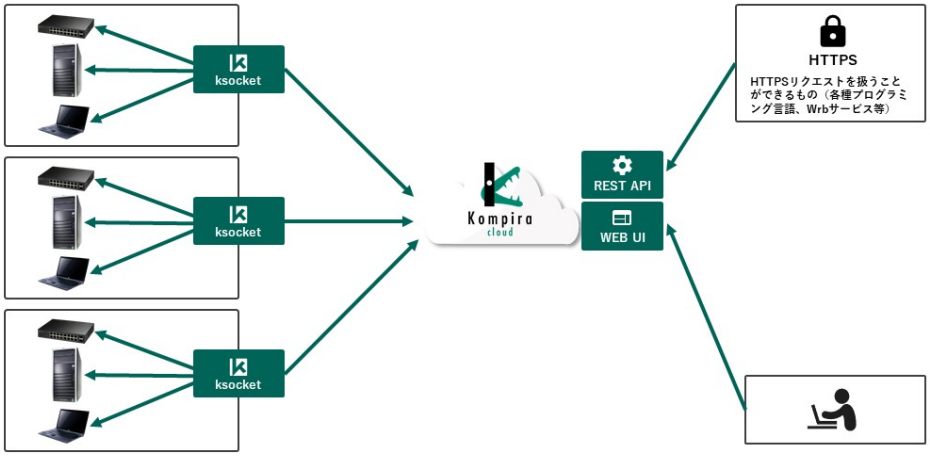
ksocket は以下のような特徴を備えています。

- 調査対象に対するアプリケーションのインストールが不要
  - 情報収集エージェントの全台導入が現実的ではない台数の環境にも対応しています。
- インストール先での依存解決が不要（例: `apt`, `yum` によるライブラリのインストールなど）
  - OS のライブラリに依存しないため、導入マシンの環境を壊しません。

ksocket によって収集された情報は **Kompira cloud** に送信されます。 収集された情報は **Kompira cloud** で提供されている Web UI および REST API を介して閲覧することができます。

分断された複数のネットワークの情報を収集したい場合は、 **ksocket** を各ネットワークごとにインストールし、 **Kompira cloud** でまとめて管理することも可能です。

以下は分断された三つのネットワークを横断的に管理する場合の構成概念図です。



図にある通り、分断された各ネットワークにそれぞれ **ksocket** を一つ導入することで、分断されたネットワークの横断的な調査が可能となります。

注釈: このドキュメントに利用されている画像は開発中のものを含むため、お使いの環境と若干異なる場合があります。あらかじめご了承ください。

第 1 章

環境情報

本ドキュメントは以下のバージョン以降を想定しています。

アプリケーション	バージョン	ファイル名
ksocket-core	1.8.0	`ksocket-linux-64-<release>.<build>.tar.gz` (Linux) `ksocket-win-64-<release>.<build>.zip` (Windows)
ksocket-networks	1.7.0	上記ファイルに同封

ksocket は以下の環境での動作をサポートしています。

- CentOS 6 (amd64)
- CentOS 7 (amd64)
- Ubuntu Server 16.04 (amd64)
- Ubuntu Server 18.04 (amd64)
- Windows Server 2012 (amd64)
- Windows Server 2012 R2 (amd64)
- Windows Server 2016 (amd64)

第 2 章

利用方法

2.1 インストール

ksocket は単一ファイルのインストーラにて提供され、対象マシンにて実行することでインストールできます。インストーラーは `ksocket-linux-64-<release>.<build>.tar.gz` もしくは `ksocket-win-64-<release>.<build>.zip` というファイルで提供され、ファイル名の `<release>` および `<build>` は、それぞれリリース番号およびビルド番号で置き換えられます。

注釈: インストールの途中で *ksocket トークン* が必要になります。Kompira cloud blog の ksocket トークンの発行 を参考に事前に有効な ksocket トークンを発行しておいてください。

なお、インストール時に適当な値を入力することで、スキップすることも可能ですが、その場合はインストール後に 設定ファイル に適切な値を指定した上で **ksocket** を再起動してください。

警告: 以前のバージョンの ksocket が存在する場合は 設定ファイル や 接続情報ファイル をバックアップ後に、インストール済みの **ksocket** をアンインストールをしてから、新規インストールをしてください。

2.1.1 Linux

提供された `ksocket-linux-64-<release>.<build>.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 zxf ksocket-linux-64-<release>.<build>.tar.gz
% ./ksinstall
```

上記により、以下のそれぞれの項目に対してプロンプトが表示されるため、お使いの環境に合わせて設定してください。

1. ksocket のインストール先ディレクトリを入力してください。
2. Host - `<space_name>.cloud.kompira.jp` と入力してください。

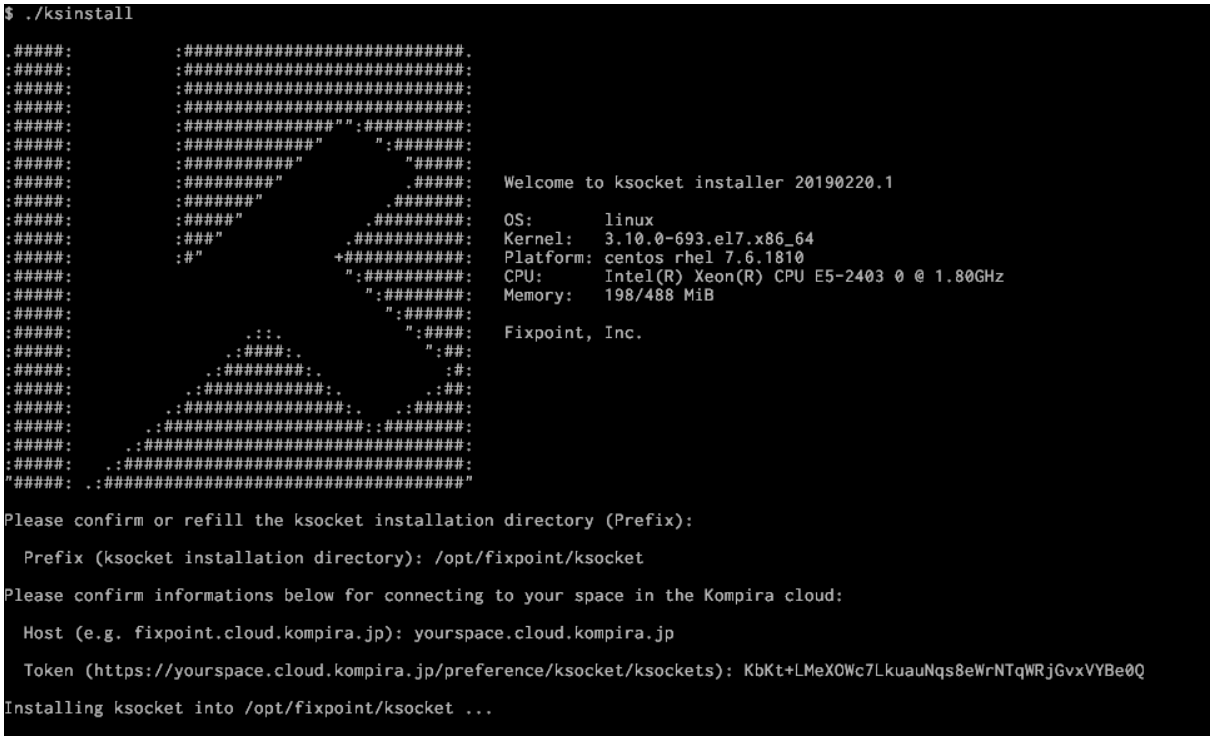


図 2.1　Linux 上にて ksocket インストーラーを起動した場合

3. ksocket token - 事前に発行した有効な ksocket トークンを記載してください。

上記を入力するとインストールが開始されます。インストール完了後 Upstart もしくは systemd が自動検出できた場合は以下のディレクトリに設定ファイルをリンクし、サービス（デーモン）の登録・起動を行います（systemd の場合は OS によりディレクトリが異なります）。

表 2.1　設定ファイルのインストール先

ソフトウェア	設定ファイルインストール先候補
Upstart	`etc/init/`
systemd	`/usr/lib/systemd/system/` (主に CentOS/RedHat)
	`/lib/systemd/system/` (主に Ubuntu)
	`usr/local/lib/systemd/system/` (FreeDesktop 系)

2.1.2 Windows

ksocket は Windows にてパケットを扱うために Npcap が提供する API に依存しています。その為事前に Npcap (0.99-r3 以上) を **WinPcap API-compatible mode** にてインストールしておく必要があります。

その後、提供された `ksocket-win-64-<release>.<build>.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、以下のそれぞれの項目に対してプロンプトが表示されるため、お使いの環境に合わせて設定してください。

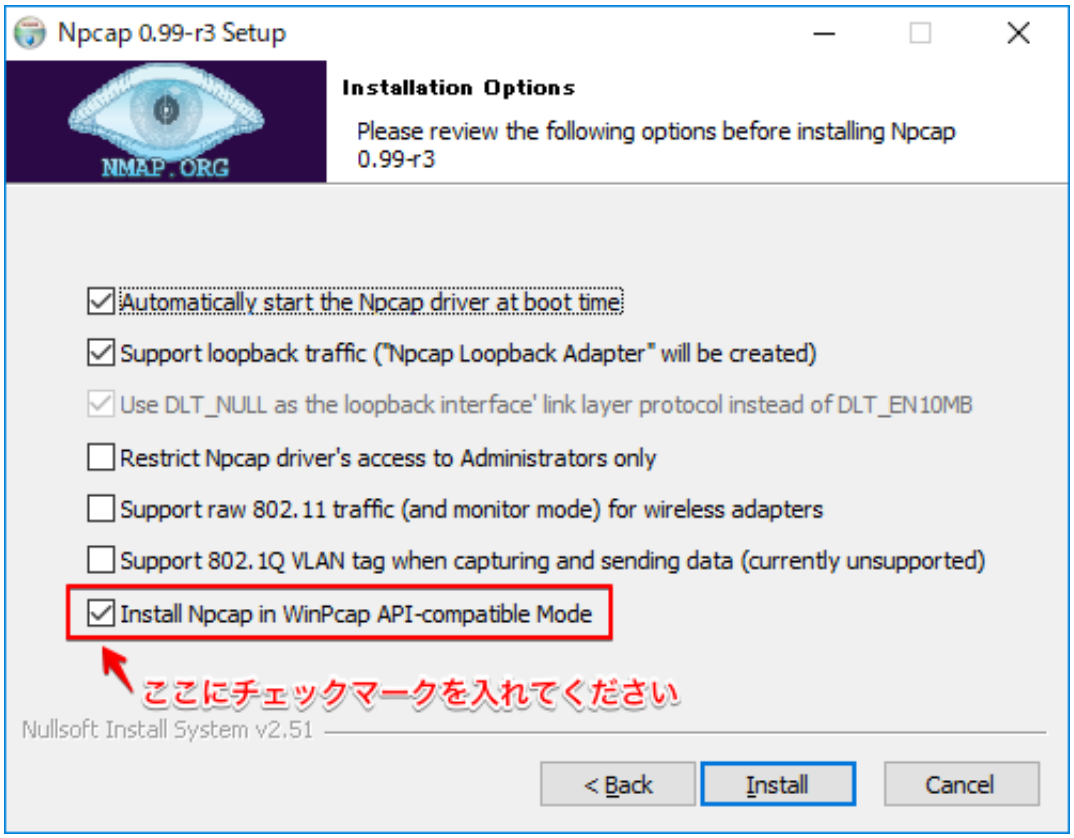


図 2.2　Npcap (0.99-r3) の WinPcap API-compatible mode

1. ksocket のインストール先ディレクトリを入力してください。
2. Host - `<space_name>.cloud.kompira.jp` と入力してください。
3. ksocket token - 事前に発行した有効な ksocket トークンを記載してください。

警告: 利用している API のバージョンの都合により ksoperations-networks 1.3.0 以降では WinPcap や Win10Pcap では動作しなくなりました。以前のバージョンにて Npcap の代わりに WinPcap や Win10Pcap を利用していた場合は、アンインストール後に Npcap をインストールしてください。

インストール完了後 `Fixpoint ksocket service` という Windows サービスが追加され、インストール中に入力された情報に従ってサーバーに接続します。またシステムの環境変数に `KSOCKET_HOME` という値を追加します。この値はサービスの起動に利用するため直接修正や削除は行わないでください。

2.2 アンインストール

ksocket はホスト OS のパッケージ等を利用しないため、以下の手順によって簡単にアンインストールが可能です。

警告: ksocket をアンインストールすると入力した設定ファイルや接続情報ファイル等も削除されるため、必要に応じてバックアップを行ってください。バックアップが必要なディレクトリは ksocket インストール先の以下 2 つです。

- `etc/ksocket` - 設定ファイル等
- `var/log/ksocket` - ログファイル等

2.2.1 Linux

以下のコマンドにより ksocket をアンインストールすることができます。

```
# /opt/fixpoint/ksocket にインストールした場合
% cd /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        # <--- y を入力するとアンインストールを実行
```

バージョン release-20190322.0 で追加: `ksuninstall` でアンインストールができるようになりました。

2.2.2 Windows

スタートメニューより「Uninstall ksocket」を実行してください。これにより Fixpoint ksocket service という Windows サービスもアンインストールされます。

2.3 接続確認

Kompira cloud にアクセスし「全体設定 > ksocket」を表示後、対象の **ksocket** を選択してください。接続状況が「接続済」となっていれば **ksocket** は正常に **Kompira cloud** に接続できています。

接続状況が「接続済」にならない場合は *よくある質問 (FAQ)* を確認してください。

2.4 バージョン確認

ksocket コマンドを直接実行することで、内部バージョン番号を確認することが出来ます。

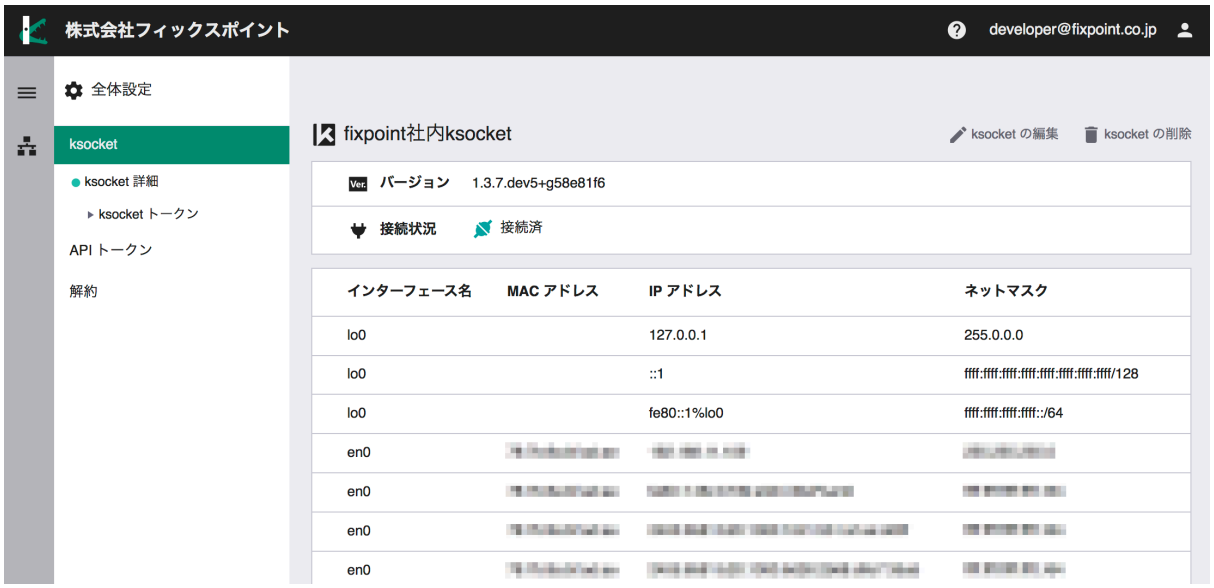


図 2.3　**ksocket** が接続されている状態の詳細画面

2.4.1 Linux

ターミナルにて以下のコマンドを実行することにより、バージョンを表示することができます。

```
% /opt/fixpoint/ksocket/bin/ksocket version
ksocket-core: 1.8.0
ksocket-networks: 1.7.0
```

2.4.2 Windows

スタートメニューより「ksocket (PowerShell)」を選択し、PowerShell コンソール上で以下のコマンドを実行することにより、バージョンを表示することができます。

```
> ksocket version
ksocket-core: 1.8.0
ksocket-networks: 1.7.0
```

2.5 再起動

2.5.1 Linux

利用しているデーモンにより、それぞれ方法が異なります。以下を参考に、お使いの環境に合わせたコマンドを利用してください。

```
# Upstart (RHEL 6.x, CentOS 6.x, Ubuntu 14.04 LTS)
% restart ksocket
```

```
# systemd (RHEL 7.x, CentOS 7.x, Ubuntu 16.04 LTS)
% systemctl ksocket restart
```

2.5.2 Windows

スタートメニューより「サービス」を起動し「Fixpoint ksocket service」を選択して再起動してください。

2.6 詳細情報取得

2.6.1 Linux (SSH)

ksocket は存在確認が取れた IP アドレスに対して SSH による詳細情報取得を試みます。そのため、スキャン対象の Linux マシンにて SSH 接続が有効化されていれば、以下の情報の取得を試みます。

- OS 名称
- OS バージョン
- RPM/deb パッケージ一覧

接続情報の記載方法は *SSH* を参照してください。

2.6.2 Windows (WinRM)

ksocket は存在確認が取れた IP アドレスに対して WinRM による詳細情報取得を試みます。そのため、スキャン対象の Windows マシンにて WinRM 接続が有効化されていれば、以下の情報の取得を試みます。

- OS 名称
- OS バージョン
- シリアル番号
- WMI パッケージ一覧
- 適用済み Windows アップデート一覧

なお、対象 Windows にて WinRM を有効化するには以下の手順を対象の Windows PowerShell コンソール (管理者権限) で実行してください。

```
# 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
# 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
```

接続情報の記載方法は *WinRM* を参照してください。

2.6.3 ネットワーク機器 (SNMP)

ksocket は存在確認が取れた IP アドレスに対して SNMP による詳細情報取得を試みます。そのため、スキャン対象のネットワーク機器にて SNMP が有効化されていれば、以下の情報の取得を試みます。

- 機種名
- バージョン
- シリアル番号

接続情報の記載方法は *SNMP* を参照してください。

2.7 AWS 上での利用

AWS の EC2 インスタンスを Kompira cloud で管理する場合、 管理対象となる VPC に対して **ksocket** がインストールされた EC2 インスタンスを用意します。**ksocket** の EC2 インスタンスに対するインストールは、 インストール と同様に行いますが、追加で本章で説明する設定を行う必要があります。

注釈: **ksocket** はスキャン対象 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 EC2 インスタンス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と同様に **ksocket** に対して適切な 接続情報ファイル を提供し SSH/WinRM 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。

2.7.1 AWS リソースに対するアクセス権限の付与

AWS の EC2 インスタンス上にインストールされた **ksocket** は、 AWS の REST API を用いて情報収集を行います。その際 **ksocket** は、 AWS リソースに対して ReadOnlyAccess 相当の権限を必要とします。以下に記載する方法から 1 つを実施してください。

IAM ロールを使用する方法

IAM ロール を使用し、 **ksocket** が AWS REST API を読み取り権限で実行できるように設定します。具体的な設定手順はリンク先を参照してください。

IAM ユーザーまたはルートユーザーのアクセスキーを使用する方法

IAM ユーザーまたはルートユーザーのアクセスキー を使用し、**ksocket** が AWS REST API を読み取り権限で実行できるように設定します。アクセスキーを発行 し、 Boto3 の Credentials ファイル[*1] を作成してください。

本ドキュメント時点では、 **ksocket** は default プロファイルを参照します。default プロファイルには、必ずデフォルトリージョンを設定してください。

awscli を使う場合は、 awscli のインストール後に `aws configure` コマンドを実行し、設定を行ってください。その際、 「Default output format」 の設定値は、 **ksocket** の動作に影響を与えません。

2.8 Azure 上での利用

Azure の Virtual Machine (VM) インスタンスを Kompira cloud で管理する場合、管理対象となる Virtual Network (VNET) に対して **ksocket** がインストールされた VM インスタンスを用意します。**ksocket** の VM インスタンスに対するインストールは、 インストール と同様に行いますが、追加で本章で説明する設定を行う必要があります。

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。

[*1] Boto3 Docs - Credentials

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と同様に **ksocket** に対して適切な 接続情報ファイル を提供し SSH/WinRM 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。

2.8.1 Azure リソースに対するアクセス権限の付与

Azure の VM インスタンス上にインストールされた **ksocket** は、 Azure の REST API を用いて情報収集を行います。その際 **ksocket** は、 Azure のサブスクリプションに対して Reader 相当の権限を必要とします。

以下に記載する方法から 1 つを実施してください。

Managed Service Identity を使用する方法

Managed Service Identity を使用し、 **ksocket** が Azure リソースにアクセスできるように設定します。

Azure ポータルを使用し、サブスクリプションをスコープとするロールを VM インスタンスに対して付与します。[*2] 具体的な手順は、 Linux VM 向けの公式ドキュメントまたは Windows VM 向けの公式ドキュメントを参照してください。

注釈: Service Principal 認証用のファイルが存在する場合は削除してください。

Service Principal を使用する方法

Service Principal を使用し、 **ksocket** が Azure リソースにアクセスできるように設定します。

Azure ポータルを使用し、Service Principal を作成 した後、サブスクリプションをスコープとするロールを Service Principal に対して付与します。[*2]

続いて、 VM インスタンス上に `$KSOCKET_HOME/etc/ksocket/azure.toml` を作成し、 TOML 形式で以下のように設定してください。

```
# 作成した Service Principal の「アプリケーション ID」
clientId = "Application ID"

# 作成した Service Principal 「認証キー」
secret = "xxxxxxxxxxxxx"
```

[*2] https://docs.microsoft.com/ja-jp/azure/role-based-access-control/role-assignments-portal#grant-access

```
# Service Principal を作成した Azure AD の「テナント ID」
tenant = "Directory ID"
```

バージョン ksocket-1.6 で変更: ファイルフォーマットを YAML 形式から TOML 形式に変更しました。

第 3 章

設定

ksocket は各種設定ファイルを TOML 0.4.0 形式 または YAML 1.1 形式で記載します。YAML はインデントの個数によりブロックを表現するフォーマットのため、下記の設定を適用する際は空白も含めるようご注意ください。

バージョン ksocket-1.6 で変更: TOML 形式での設定ファイルの記載に対応しました。これまでの YAML 形式で記載された設定ファイルにも対応していますが、新しくファイルを作成するときは TOML 形式で記載するようにしてください。

注釈: **ksocket** が扱うファイルは UTF-8 でエンコードされている必要があります

3.1 デフォルト保存先一覧

パス	説明
`$KSOCKET_HOME/etc/ksocket/ksocket.toml`	**ksocket** の設定を記載するための TOML ファイル
`$KSOCKET_HOME/etc/ksocket/ksocket.yml`	**ksocket** の設定を記載するための YAML ファイル (旧 ksocket での設定方式。deprecated)
`$KSOCKET_HOME/etc/ksocket/credentials`	リモート接続時に利用する接続ファイルのデフォルト探索ディレクトリ

なお表中の `$KSOCKET_HOME` は **ksocket** のインストールディレクトリを指します。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。

3.2 設定ファイル

ksocket は `$KSOCKET_HOME/etc/ksocket/ksocket.toml` に設定を記載します。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。

```
# ログファイルの保存先
# '-' を指定すると標準エラーに出力する
logfile = "$KSOCKET_HOME/var/log/ksocket/ksocket.log"

# ログレベルの指定。以下の値が指定可能
# - DEBUG
# - INFO
# - WARNING
# - ERROR
# - CRITICAL
loglevel = "DEBUG"

# ログ出力フォーマットの指定
# https://docs.python.jp/3/library/logging.html#logrecord-attributes
# に列挙されるフォーマット文字列を利用可能
#logformat = "%(asctime)s  %(levelname)s  %(name)s:%(funcName)s:%(lineno)d
↪%(message)s"

[connect]
# ksocket トークンの指定
token = "xxxxxxxx"

# Websocket プロトコルの指定 [wss or ws]
protocol = "wss"

# 接続先の Kompira cloud スペース URL
host = "<space_name>.cloud.kompira.jp"

# 接続に使用するポート番号
port = 443

[directories]
# 接続情報を検索するディレクトリ
#credentials = "$KSOCKET_HOME/etc/ksocket/credentials"
```

$KSOCKET_HOME/etc/ksocket/ksocket.toml.skeleton をコピーしてご利用ください。

注釈: ksocket.toml を編集した場合、反映させるためには **ksocket** の再起動を行う必要があります。

3.2.1 設定ファイル (YAML 形式での設定)

ksocket は、 設定ファイル の ksocket.toml ファイルが存在しなかったとき、*$KSOCKET_HOME*/etc/ksocket/ksocket.yml から設定を読み込みます。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。

```
# ログファイルの保存先
# '-' を指定すると標準エラーに出力する
logfile: ${KSOCKET_HOME}/var/log/ksocket/ksocket.log
```

```
# ログレベルの指定。以下の値が指定可能
# - DEBUG
# - INFO
# - WARNING
# - ERROR
# - CRITICAL
loglevel: 'DEBUG'

# ログ出力フォーマットの指定。
# https://docs.python.jp/3/library/logging.html#logrecord-attributes
# に列挙されるフォーマット文字列を利用可能
#logformat: "%(asctime)s  %(levelname)s  %(name)s:%(funcName)s:%(lineno)d
↪%(message)s"

# connect コマンドの設定
connect:
  # ksocket トークンの指定
  token: 'xxxxxxxxxx'

  # 接続先のホスト名
  host: '<space_name>.cloud.kompira.jp'

  # 接続先のポート番号
  port: 443

# ディレクトリ関係
directories:
  # 接続情報を検索するディレクトリ
  credentials: $KSOCKET_HOME/etc/ksocket/credentials
```

バージョン ksocket-1.6 で撤廃: `ksocket.yaml` の代わりに `ksocket.toml` で設定を行うようにしてください。

注釈: `ksocket.yml` を編集した場合、反映させるためには **ksocket** の再起動を行う必要があります。

3.3 接続情報ファイル

$KSOCKET_HOME`/etc/ksocket/credentials` 以下には、リモートホストに対する接続情報を指定します。接続情報には対象とする IP アドレス、利用するアカウント、ポート番号等が含まれ、接続方式ごとにファイルを作成します。

3.3.1 ファイルフォーマット

本項では SSH (Secure Shell), WinRM (Windows Remote Management), SNMP (Simple Network Management Protocol) それぞれの接続情報の記載方法を解説します。

SSH

SSH 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。このフォーマットのファイルは *$KSOCKET_HOME*/etc/ksocket/credentials/ssh 以下に保存してください。

バージョン ksocket-1.6 で変更: TOML 形式での設定ファイルの記載に対応しました。これまでの YAML 形式で記載された設定ファイルにも対応していますが、新しくファイルを作成するときは TOML 形式で記載するようにしてください。

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)
#
#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。

SSH (YAML 形式での設定)

```
# 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 10.10.0.0/16 のホスト及び 10.20.0.1, 10.20.0.3 が指定されている
includes:
  - 10.10.0.0/16
  - 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:
  - 10.10.1.0/24
  - 10.20.0.1

# 接続に使用するアカウント情報
account:
  username: john
  password: Passw0rd

  # 接続に使用する鍵ファイル候補一覧
  clientKeys:
    - filename: ../../client_keys/id_john
      passphrase: helloworld
    - filename: ../../client_keys/id_rsa.common
      passphrase: common

# 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
sshTunnels:
  - 10.10.0.1
  - 10.10.0.2
```

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

バージョン ksocket-1.6 で撤廃: TOML 形式で設定を行うようにしてください。

WinRM

WinRM 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。このフォーマットのファイルは *$KSOCKET_HOME*`/etc/ksocket/credentials/winrm` 以下に保存してください。

バージョン ksocket-1.6 で変更: TOML 形式での設定ファイルの記載に対応しました。これまでの YAML 形式で記載された設定ファイルにも対応していますが、新しくファイルを作成するときは TOML 形式で記載する

ようにしてください。

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
#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。

WinRM (YAML 形式での設定)

```
# 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 10.10.0.0/16 のホスト及び 10.20.0.1, 10.20.0.3 が指定されている
includes:
  - 10.10.0.0/16
  - 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:
  - 10.10.1.0/24
  - 10.20.0.1

# 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

# 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
authMethod: ntlm

# 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: always
```

```
# 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

# 踏み台とするホストの IP アドレスリスト
sshTunnels:
  - 10.10.0.1
  - 10.10.0.2
```

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。

バージョン ksocket-1.6 で撤廃: TOML 形式で設定を行うようにしてください。

SNMP

SNMP 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。このフォーマットのファイルは *$KSOCKET_HOME*`/etc/ksocket/credentials/snmp` 以下に保存してください。

バージョン ksocket-1.6 で変更: TOML 形式での設定ファイルの記載に対応しました。これまでの YAML 形式で記載された設定ファイルにも対応していますが、新しくファイルを作成するときは TOML 形式で記載するようにしてください。

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)
# 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"
```

SNMP (YAML 形式での設定)

```
# 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:
  - 10.10.1.0/24
  - 10.20.0.1

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
  # SNMPv3 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)
  # 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


# 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
```

バージョン ksocket-1.6 で撤廃: TOML 形式で設定を行うようにしてください。

3.3.2 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_HOME*/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
   （a）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
   （a）ファイル名が .toml または .yml で終わっていない場合は除外
   （b）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
   （c）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
   （d）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。

3.3.3 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。

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

トラブルシューティング

4.1 サポート

ksocket 上で問題が発生した場合、 設定ファイル に記載された ksocket 設定ファイルでログレベルを変更することにより、 **ksocket** の詳細なログを確認することができます。

```
loglevel = "DEBUG"
```

問題の状況と共にログファイル (ksocket.log) を添付し、 support@kompira.jp までお問い合わせください。

4.2 よくある質問 (FAQ)

ここではよくある質問や、つまづきやすいポイントなどを Q&A 方式でお答えします。

ksocket が Kompira cloud に接続されない

- **ksocket** と **Kompira cloud** 間の通信には、TCP 443 番ポートを使用します。 **ksocket** がインストールされた機器がインターネットにアクセス可能かどうかを確認してください。また、 TCP 443 番ポートでの外部に対する接続制限が設定されていないかどうかを確認してください。
- **Kompira cloud** で発行された ksocket トークンが間違っている場合、正しく接続することができません。 ksocket トークンを発行し直し、 **ksocket** に設定して接続を再試行してみてください。
- ksocket トークンには失効日があり、失効日を過ぎると **Kompira cloud** は **ksocket** からの接続を受け付けなくなります。ksocket トークンの失効日を確認してみてください。

スキャンを実行したが何も取得されない

- **ksocket** はスキャンのはじめに、 **ksocket** がインストールされた機器のデフォルトゲートウェイ IP アドレスを取得し、その IP アドレスに対して SNMP 接続を行うことで探索を開始します。もしデフォルトゲートウェイが SNMP に応答しない設定の場合、設定の変更が可能かどうか検討してみてください。

第 5 章

付録

5.1 デフォルトのインストール先

表 5.1　デフォルトのインストール先

OS	インストール先 （$KSOCKET_HOME）
Windows	C:\ProgramData\Fixpoint\ksocket
Linux	/opt/fixpoint/ksocket

第 6 章

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。

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 設定ファイル を参照。

接続情報ファイル　**ksocket** が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 接続情報ファイル を参照。

索引