
Ksocket

リリース *v2.3.1*

Fixpoint, Inc.

2022 年 11 月 25 日

目次

第 1 章	インストール	2
1.1	ローカルネットワークへのインストール	2
1.2	Azure へのインストール	6
1.3	AWS へのインストール	8
1.4	アンインストール	10
第 2 章	使用方法	12
2.1	コマンド実行方法	12
2.2	設定コマンド	13
2.3	自己診断コマンド	15
2.4	サービス管理コマンド	18
第 3 章	設定ファイル	23
3.1	Ksocket 設定	23
3.2	接続情報	25
第 4 章	トラブルシュート	33
4.1	ksocket コマンドが見つからない	33
4.2	自己診断コマンドによる診断	33
4.3	デバッグログによる調査	34
4.4	サポートへ問い合わせ	35
第 5 章	変更履歴	36
5.1	Ksocket v2.3.1	36
5.2	Ksocket v2.3.0	37
5.3	Ksocket v2.2.4	38
5.4	Ksocket v2.2.1	39
5.5	Ksocket v2.2.0	39
5.6	Ksocket v2.1.1	40
5.7	Ksocket v2.1.0	41
5.8	Ksocket v2.0.1	42
5.9	Ksocket v2.0.0	43
第 6 章	付録	45
6.1	デフォルトのインストール先	45
6.2	WinRM 接続の有効化	45
6.3	用語集	46
6.4	Ksocket v1 から v2 への移行	46
	索引	49

Ksocket は [株式会社フィックスポイント](https://www.fixpoint.co.jp/)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](https://cloud.kompira.jp/)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](https://support.kompira.jp/2017/01/01/ksocket/)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://support.kompira.jp/2017/01/01/ksocket/>

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 へのインストール

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://support.kompira.jp/2017/01/01/ksocket/>

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。

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、比較的处理能力が低い 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。

1.1.2 Windows へのインストール

Npcap のインストール

Ksocket は Windows にてパケットを扱うために Npcap⁷ 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。

⁷ <https://nmap.org/npcap>

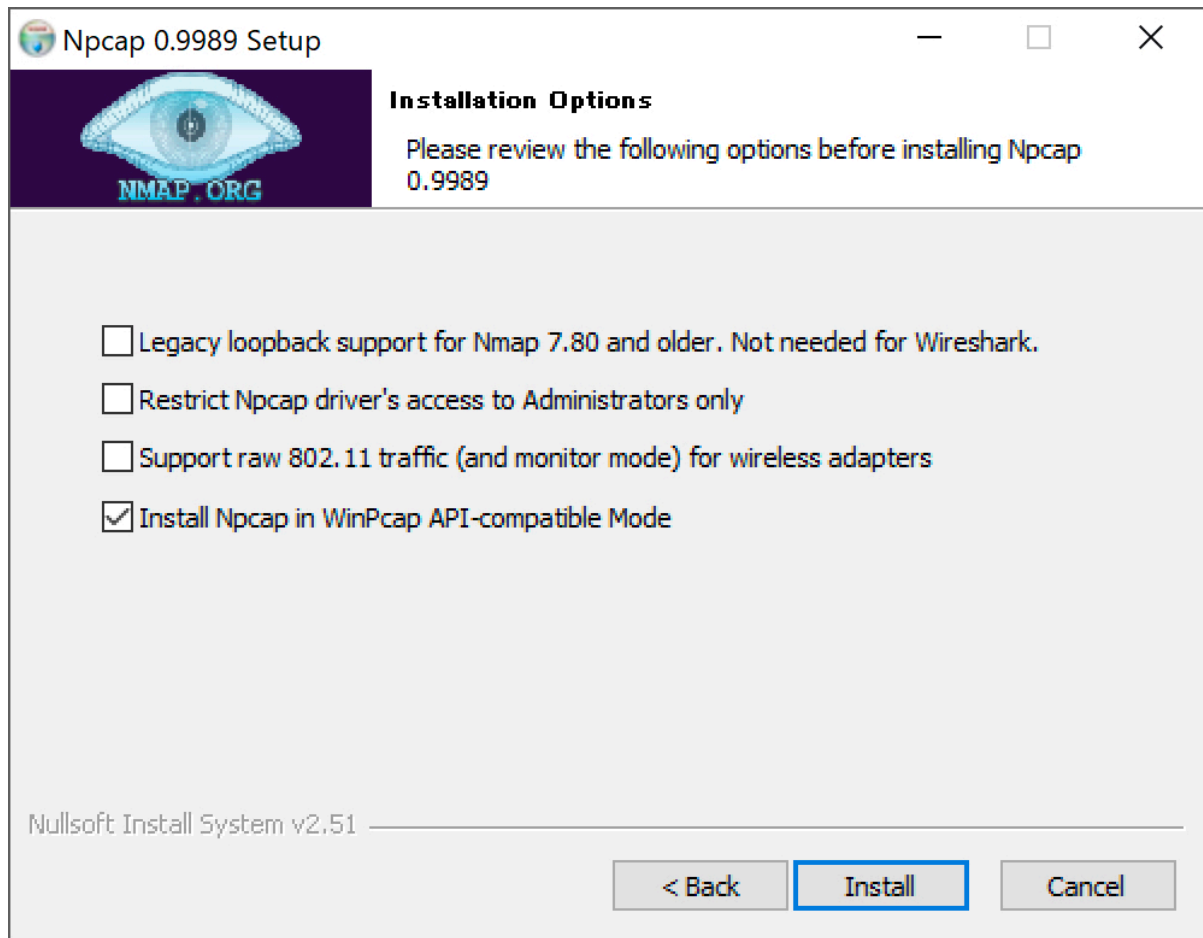


図 2: Npcap (0.9989) の WinPcap API-compatible mode

Ksocket インストーラの実行

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、ksocket インストーラーを起動します。Ksocket インストーラーの利用方法は Linux と同様になります。1.1.1 [Linux へのインストール](#) を参照し、インストールを進めてください。

インストール先フォルダーの表示

Ksocket は標準では C:\ProgramData\Fixpoint\ksocket 以下に配置されます。上記フォルダーが Windows エクスプローラー上では表示されない場合は、以下の次の手順でフォルダーオプションを変更して下さい。

1. エクスプローラーからオプションを表示します。
2. 表示タブから「隠しファイル、隠しフォルダー、および隠しドライブを表示する」を選択し、適用します

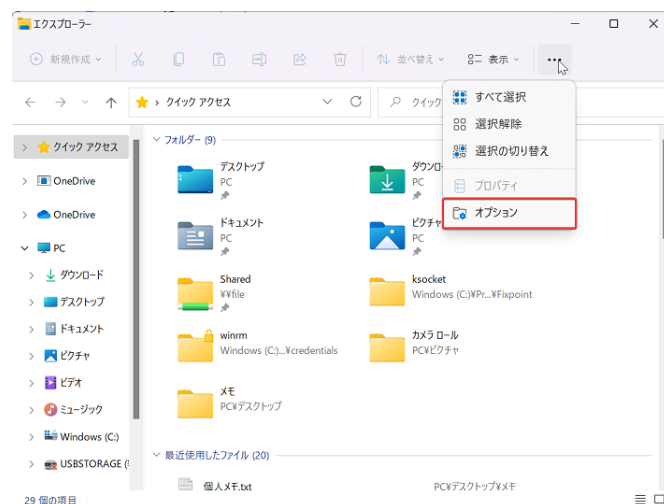


図 3: エクスプローラーのオプション

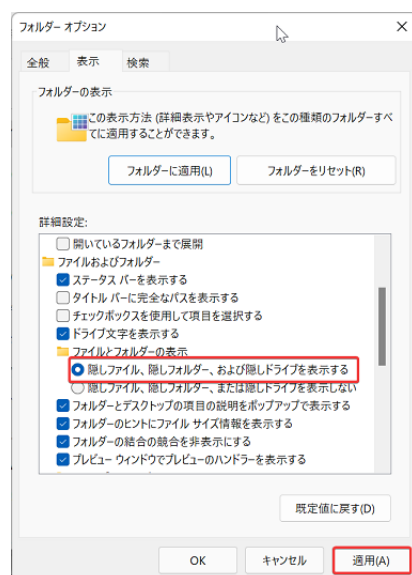


図 4: フォルダーオプションの表示タブ

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 ID⁹ 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。

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>

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。

```
# 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 = "xxxxxxxxxxxxxxxx"

# 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。

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>

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 [ReadOnlyAccess¹⁵](#) 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。

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

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。

- [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:
```

(次のページに続く)

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

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

(前のページからの続き)

- ```
- 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
```

## 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 ( オプション )

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 サービス活用方法<sup>21</sup>](#)
  - [ローカルネットワークのスキャン<sup>22</sup>](#)
  - [ローカルネットワークスキャンの実行例<sup>23</sup>](#)
  - [ローカルネットワークスキャンの仕組み<sup>24</sup>](#)
- 

### 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:

```
% ksocket version
2.0.0
```

---

<sup>21</sup> [https://support.kompira.jp/column\\_category/allarticlesonhowtousetheservice/](https://support.kompira.jp/column_category/allarticlesonhowtousetheservice/)

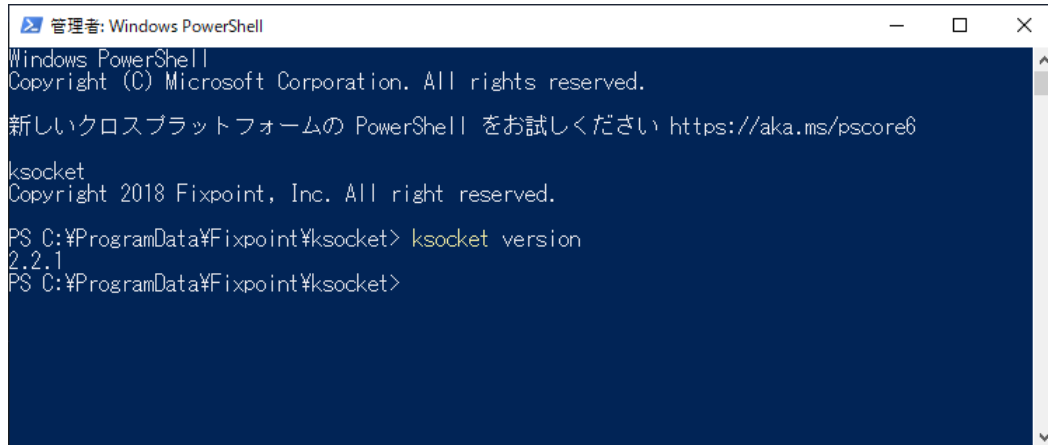
<sup>22</sup> <https://support.kompira.jp/column/scanlocalnetwork/>

<sup>23</sup> <https://support.kompira.jp/column/localnetwork-example/>

<sup>24</sup> <https://support.kompira.jp/column/howlocalnetworkscanworks/>

## 2.1.2 Windows

スタートメニューから スタート > *Fixpoint* > *Ksocket(PowerShell)* を起動し、以下のコマンドが実行できることを確認してください。



```
管理者: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

新しいクロスプラットフォームの PowerShell をお試しください https://aka.ms/pscore6

ksocket
Copyright 2018 Fixpoint, Inc. All right reserved.

PS C:\ProgramData\Fixpoint\ksocket> ksocket version
2.2.1
PS C:\ProgramData\Fixpoint\ksocket>
```

## 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
```

(次のページに続く)

(前のページからの続き)

```
connect.resumePlateau=10
connect.resumeThreshold=120.0
connect.requestTimeout=60.0
connect.requestRetries=10
connect.maxOperations=100
connect.maxRunningOperations=5
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 diagnostics` 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 diagnostics` 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 diagnostics 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 diagnostics system` を呼び出してください。以下のようになれば OK であれば、システム的な問題はありません:

```
% ksocket diagnostics 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
```

(次のページに続く)

(前のページからの続き)

```
ntlm_auth OK
jsonref OK
stringcase OK
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 diagnostics 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 diagnostics 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 diagnostics 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、疎通状況に問題はなりません:

```
% ksocket diagnostics 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
```

(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: /opt/fixpoint/ksocket/etc/ksocket/ksocket.toml

$PYTHONHOME:
$PYTHONPATH:
$PYTHONOPTIMIZE:
$PYTHONENCODING:
$PYTHONUTF8:
$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 diagnostics 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 diagnostics 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

```

(次のページに続く)

(前のページからの続き)

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

KSOCKET_INSECURE: False

Host: fixpoint.cloud.kompira.jp
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](#)<sup>25</sup>, [Upstart](#)<sup>26</sup> もしくは [NSSM](#)<sup>27</sup> 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)

#### Linux

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
```

(次のページに続く)

<sup>25</sup> <https://www.freedesktop.org/wiki/Software/systemd/>

<sup>26</sup> <http://upstart.ubuntu.com/>

<sup>27</sup> <https://nssm.cc/>

(前のページからの続き)

```
Get status of the service OK
Restart the service OK
```

Linux では [systemd](#)<sup>28</sup> もしくは [Upstart](#)<sup>29</sup> 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**

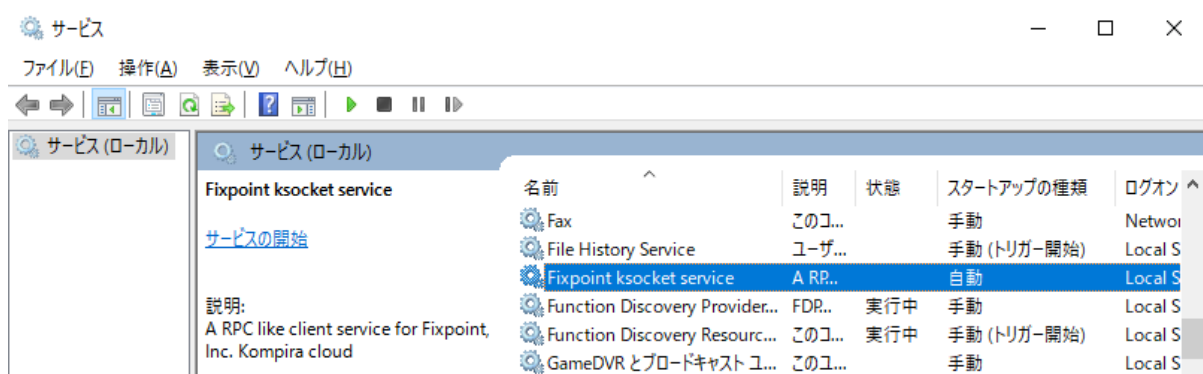
```
/usr/lib/systemd/system/ (CentOS/RedHat) /lib/systemd/system/ (Ubuntu/Debian)
/usr/local/lib/systemd/system/ (FreeDesktop)
```

**Upstart**

```
/etc/init
```

**Windows**

Windows では Ksocket にバンドルされている [NSSM](#)<sup>30</sup> 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。サービス名は Fixpoint ksocket service になります。



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。

<sup>28</sup> <https://www.freedesktop.org/wiki/Software/systemd/>

<sup>29</sup> <http://upstart.ubuntu.com/>

<sup>30</sup> <https://nssm.cc/>

## 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)

### Linux

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
```

### Windows

スタート > Windows 管理ツール > サービス を開き、Fixpoint ksocket service から状態を確認することができます。

| サービス (ローカル)                    |                    |     |             |                 |  |
|--------------------------------|--------------------|-----|-------------|-----------------|--|
| 名前                             | 説明                 | 状態  | スタートアップの種類  | ログオン            |  |
| Fixpoint ksocket service       |                    |     |             |                 |  |
| Fax                            | このコンピューター...       |     | 手動          | Network Service |  |
| File History Service           | ユーザー ファイル...       |     | 手動 (トリガー開始) | Local System    |  |
| Fixpoint ksocket service       | A RPC like clie... | 実行中 | 自動          | Local System    |  |
| Function Discovery Provider... | FDPHOST サー...      | 実行中 | 手動          | Local Service   |  |
| Function Discovery Resourc...  | このコンピューター...       | 実行中 | 手動 (トリガー開始) | Local Service   |  |
| GameDVR とブロードキャストユ...          | このユーザー サー...       |     | 手動          | Local System    |  |
| Geolocation Service            | このサービスは、...        | 実行中 | 手動 (トリガー開始) | Local System    |  |
| Google Chrome Elevation S...   |                    |     | 手動          | Local System    |  |
| Google Update サービス (gup...     | Google ソフトウ...     |     | 自動 (遅延開始)   | Local System    |  |
| Google Update サービス (gup...     | Google ソフトウ...     |     | 手動          | Local System    |  |
| GraphicsPerfSvc                | Graphics perfo...  |     | 手動 (トリガー開始) | Local System    |  |

## 2.4.4 サービスの開始 (start)

### Linux

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
```

### Windows

スタート > Windows 管理ツール > サービス を開きます。Fixpoint ksocket service を右クリックでメニューを表示し、開始 をクリックしてください。

## 2.4.5 サービスの停止 (stop)

### Linux

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
```

### Windows

スタート > *Windows* 管理ツール > サービス を開きます。Fixpoint ksocket service を右クリックでメニューを表示し、停止 をクリックしてください。

## 2.4.6 サービスの再起動 (restart)

### Linux

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。

### Windows

スタート > *Windows* 管理ツール > サービス を開きます。Fixpoint ksocket service を右クリックでメニューを表示し、再起動 をクリックしてください。



## 第3章 設定ファイル

Ksocket は各種設定ファイルを TOML<sup>31</sup> 0.4.0 形式で記載します。

---

注釈: 後方互換性のため YAML<sup>32</sup> 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 \$KSOCKET\_HOME/etc/ksocket/ksocket.toml に記載された設定を読み込みます。ここで \$KSOCKET\_HOME は以下になります。

#### Windows

C:\ProgramData\Fixpoint\ksocket

#### Linux

/opt/fixpoint/ksocket

設定の確認 / 変更には 2.2 設定コマンド が利用できます。

---

注釈: Windows では設定ファイルの編集に メモ帳 が利用できますが、その際には 管理者として実行 するよう注意してください。

---

**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。

---

<sup>31</sup> <https://github.com/toml-lang/toml>

<sup>32</sup> <https://yaml.org/>

### 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

### 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<sup>33</sup>](#) 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。

---

<sup>33</sup> <https://support.kompira.jp/column/issuanceofksockettokens/>

### 3.1.6 connect.protocol

Kompira cloud との接続時に使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

### 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<sup>34</sup>](#) 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。

<sup>34</sup> <https://support.kompira.jp/column/howlocalnetworkscanworks/>

### 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)
#
#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。

注釈: Windows では、以下に注意してください。

- ディレクトリの指定では¥(又は\ )の代わりに / を利用してください
- 例 `c:¥ProgramData¥fixpoint` ではなく `c:/ProgramData/fixpoint` として下さい
- ディレクトリは絶対パス、又は `C:\ProgramData\Fixpoint\ksocket\etc\ksocket\credentials\ssh\` からの相対パスで指定します
- 秘密鍵の配置場所に制限はありませんが、管理・運用上の理由で接続情報と同一ディレクトリをお勧めします

### 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
#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
```

(次のページに続く)

(前のページからの続き)

```
#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"
```

(次のページに続く)

(前のページからの続き)

```
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"
```

### 3.2.4 VMware vSphere API (vCenter、ESXi に対しての接続)

VMware vSphere API 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/vsphere` 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 443
#port = 443

SSL 証明書の検証の無効化。省略時は true
#disableSslCertValidation = true

接続のタイムアウト時間
#timeout = 5.0

接続に使用するアカウント情報
```

(次のページに続く)



(前のページからの続き)

```
[account]
ユーザ名
username = "john"

パスワード
password = "passw0rd"
```

### 3.2.5 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 `.yaml` 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。

### 3.2.6 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
 +- _special.toml
 +- 02_final.toml
```

(次のページに続く)

(前のページからの続き)

```
+ 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

システムのな不備があります。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](#)<sup>35</sup> 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
```

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。

---

<sup>35</sup> <https://support.kompira.jp/column/useksocketviaproxy/>

## 4.4 サポートへ問い合わせ

Fixpoint コミュニティサイト<sup>36</sup> の Sonar 関連トピックもしくは [support@kompira.jp](mailto: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

---

<sup>36</sup> <https://kompira.zendesk.com/hc/ja/community/topics>

## 第5章 変更履歴

### 5.1 Ksocket v2.3.1

リリース日

2022/11/30

#### 5.1.1 新機能

なし

#### 5.1.2 変更点

なし

#### 5.1.3 削除予定の機能

なし

#### 5.1.4 削除された機能

なし

#### 5.1.5 バグ修正

- 一部の Linux ディストリビューションにおいて、Ksocket の実行に依存ライブラリの libpcap.so を別途インストールする必要が生じていていた問題を修正

### 5.1.6 脆弱性対応

なし

## 5.2 Ksocket v2.3.0

リリース日

2022/11/09

### 5.2.1 新機能

- VMware vSphere API を使用した vCenter と ESXi の情報取得をサポート

### 5.2.2 変更点

なし

### 5.2.3 削除予定の機能

なし

### 5.2.4 削除された機能

なし

### 5.2.5 バグ修正

なし

### 5.2.6 脆弱性対応

なし

## 5.3 Ksocket v2.2.4

リリース日

2022/10/05

### 5.3.1 新機能

なし

### 5.3.2 変更点

- Ksocket のサポート対象 OS について、正規サポートが終了している下記 3 OS のサポートを終了
  - RHEL 6
  - CentOS 6
  - Ubuntu 16.04

### 5.3.3 削除予定の機能

なし

### 5.3.4 削除された機能

なし

### 5.3.5 バグ修正

- 設定ファイルのフォーマットが不正な状態で診断コマンドを実行した場合、診断自体がエラーで止まっていた問題を修正

### 5.3.6 脆弱性対応

なし



## 5.4 Ksocket v2.2.1

リリース日

2020/12/17

### 5.4.1 新機能

なし

### 5.4.2 変更点

なし

### 5.4.3 削除予定の機能

なし

### 5.4.4 削除された機能

なし

### 5.4.5 バグ修正

- 特定のスキャン命令が正常に処理されていなかった不具合を修正  
2020/12 時点では Juniper 製品の LLDP 情報収集処理のみが影響

### 5.4.6 脆弱性対応

なし

## 5.5 Ksocket v2.2.0

リリース日

2020/10/19

### **5.5.1 新機能**

- ARM 64bit(ARMv8 aarch64) 対応版をリリース

### **5.5.2 変更点**

なし

### **5.5.3 削除予定の機能**

なし

### **5.5.4 削除された機能**

なし

### **5.5.5 バグ修正**

- コマンドのヘルプで利用可能コマンド一覧が表示されていなかった不具合を修正

### **5.5.6 脆弱性対応**

なし

## **5.6 Ksocket v2.1.1**

リリース日

2020/08/25

### **5.6.1 新機能**

なし

### 5.6.2 変更点

なし

### 5.6.3 削除予定の機能

なし

### 5.6.4 削除された機能

なし

### 5.6.5 バグ修正

- 一部のサンプル設定ファイルがインストール時に展開されていなかった問題を修正

### 5.6.6 脆弱性対応

なし

## 5.7 Ksocket v2.1.0

リリース日

2020/05/14

### 5.7.1 新機能

- Linux(CentOS7) 上の L2TP/IPsec(ppp) インターフェース経由でのスキャンに対応

### 5.7.2 変更点

なし

### 5.7.3 削除予定の機能

なし

### 5.7.4 削除された機能

なし

### 5.7.5 バグ修正

なし

### 5.7.6 脆弱性対応

なし

## 5.8 Ksocket v2.0.1

リリース日

2020/04/03

### 5.8.1 新機能

なし

### 5.8.2 変更点

- 自己診断コマンド *ksocket dignostics* を *ksocket diagnostics* にリネーム

### 5.8.3 削除予定の機能

なし

### 5.8.4 削除された機能

なし

### 5.8.5 バグ修正

- 自己診断コマンドにて疏通確認時にホストの SSL 証明書のみを参照していた問題を修正

### 5.8.6 脆弱性対応

なし

## 5.9 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*

### 5.9.1 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

## 5.9.2 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

## 5.9.3 削除予定の機能

- YAML 形式の設定ファイル

## 5.9.4 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

## 5.9.5 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

## 5.9.6 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<sup>37</sup>
- Ksocket 接続受け入れの一括設定マニュアル<sup>38</sup>

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
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
```

(次のページに続く)

<sup>37</sup> <https://blog.cloud.kompira.jp/entry/utility/winrm-config-script>

<sup>38</sup> <https://blog.cloud.kompira.jp/entry/manual/winrm-bulk-setting-manual>

(前のページからの続き)

```
> 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
> wmicgmt.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

Ksocket トークン, [46](#)

接続情報ファイル, [46](#)

設定ファイル, [46](#)