

## 第19回ICN研究会ワークショップ

# ICN ・ Cefore ・ cefpyco チュートリアル

---

2021年 8月26日（木） ・ 8月27日（金）

**Cefore を用いた通信**

■ TODO

**cefpyco を用いた通信**

■ TODO

**付録：インストール  
マニュアル**

■ TODO

## ■ Cefore を用いた通信

- cefnetd と csmgrd の起動・停止
- 設定と変更

## ■ cefpyco を用いた通信

- cefnetd への接続
- Interest・Data パケットの送受信
- 簡易 Consumer・Producer アプリの作成

## ■ 参考：インストールマニュアル

- Cefore の導入・環境構築
- cefpyco の導入



# Cefore の機能・ツール一覧

機能	形態	説明
cefnetd	daemon	フォワーディングデーモン
cefnetdstart	utility	フォワーディングデーモン起動ユーティリティ
cefnetdstop	utility	フォワーディングデーモン停止ユーティリティ
cefstatus	utility	cefnetdのstatus標準出力ユーティリティ
cefroute	utility	FIB操作ユーティリティ
ccninfo/cefinfo	tool	ネットワーク管理ツール
cefputfile	tool	任意のファイルをNamed Cobに変換しcefnetdへ入力する
cefgetfile	tool	cefnetdを介して取得したコンテンツをファイルとして出力する
cefgetchunk	tool	指定されたNamed Cobを取得し、ペイロードを標準出力する
cefputstream	tool	標準入力をNamed Cobに変換しcefnetdへ入力する
cefgetstream	tool	cefnetdを介して取得したコンテンツを標準出力する
csmgrd	daemon	コンテンツストア管理デーモン (CS_MODE=2)
csmgrdstart	utility	csmgrd起動ユーティリティ
csmgrdstop	utility	csmgrd停止ユーティリティ
csmgrstatus	utility	csmgrdのstatus標準出力ユーティリティ

# 1. cefnetdとcsmgrdの起動

---



# cefnetd と csmgrd の起動手順

## ① 動作確認

- (1-1) cefnetd の起動確認と停止
- (1-2) csmgrd の起動確認と停止

## ② cefore の設定ファイルの説明

- (2-1) cefnetd.conf の設定
- (2-2) cefnetd.fib の設定
- (2-3) csmgrd.conf の設定

## ③ 設定ファイルの変更

- (3-1) ルーティングテーブルの設定 (cefnetd.fib)
- (3-2) cefnetdがキャッシュを利用するように設定

# (1-1) cefnetd の起動確認

## ■ cefnetdの起動・起動確認・停止コマンド

\$ sudo cefnetdstart	# cefnetdを起動
\$ cefstatus	# cefnetdのステータスを確認
\$ sudo cefnetdstop	# cefnetdを停止

## ■ 動作例

```
cefore:~/cefore$ sudo cefnetdstart
2021-08-26 15:13:00.123 [cefnetd] INFO: [client] Config directory is /usr/local/Cefore
...
2021-08-26 15:13:00.123 [cefnetd] INFO: Not use Content Store
2021-08-26 15:13:00.456 [cefnetd] INFO: Running
cefore:~/cefore$ cefstatus
2021-08-26 15:13:05.123 [cefctrl] INFO: [client] Config directory is /usr/local/cefore
...
Port          : 9896
Rx Frames     : 0
Tx Frames     : 0
Cache Mode    : None
Faces :
  faceid =   4 : IPv4 Listen face (udp)
  faceid =   0 : Local face
  faceid =  16 : Local face
  faceid =   5 : IPv6 Listen face (udp)
  faceid =   6 : IPv4 Listen face (tcp)
  faceid =   7 : IPv6 Listen face (tcp)
FIB :
  Entry is empty
PIT :
  Entry is empty

cefore:~/cefore$ sudo cefnetdstop
2021-08-26 15:13:10.456 [cefctrl] INFO: [client] Config directory is /usr/local/Cefore
...
2021-08-26 15:13:10.789 [cefnetd] INFO: Stop
```

# (1-2) csmgrd の起動確認

## ■ csmgrdの起動・起動確認・停止コマンド

\$ sudo csmgrdstart	# csmgrdを起動
\$ csmgrstatus	# csmgrdのステータスを確認
\$ sudo csmgrdstop	# csmgrdを停止

## ■ 動作例

```
cefore:~/cefore$ sudo csmgrdstart
2021-08-26 15:14:00.123 [csmgrd] INFO: Config directory is /usr/local/cefore.
...
2021-08-26 15:14:00.123 [csmgrd] INFO: Loading csmgrd.conf ... OK
2021-08-26 15:14:00.123 [csmgrd] INFO: Running

cefore:~/cefore$ sudo csmgrstatus ccnx:/

Connect to 127.0.0.1:9799
2021-08-26 15:14:05.456 [csmgrd] INFO: Open TCP peer: 127.0.0.1:37920, socket : 5
***** Connection Status Report *****
All Connection Num          : 1

***** Cache Status Report *****
Number of Cached Contents   : 0

2021-08-26 15:14:00.456 [csmgrd] INFO: Close TCP peer: 127.0.0.1:37920

cefore:~/cefore$ sudo csmgrdstop
2021-08-26 15:14:10.123 [csmgrd] INFO: Open TCP peer: 127.0.0.1:37922, socket : 5
2021-08-26 15:14:10.123 [csmgrd] INFO: csmgrdstop from root
2021-08-26 15:14:10.789 [csmgrd] INFO: Stop
```

**※現行版(0.8.3)ではPCのメモリ4GB以上推奨  
(2GB以下の場合は起動しないでください)**



## ② cefore の設定

### ■ /usr/local/cefore に設定ファイルが存在\*1

ファイル名	説明
<b>cefnetd.conf</b>	cefnetdの設定ファイル
<b>cefnetd.fib</b>	cefnetdのFIBエントリの設定ファイル
<b>csmgrd.conf</b>	csmgrdの設定ファイル
<ul style="list-style-type: none"><li>• cefnetd.key</li><li>• ccore-public-key</li><li>• default-public-key</li><li>• default-private-key</li></ul>	InterestとContet ObjectのValidationに使用する公開鍵と秘密鍵の設定ファイル、およびデフォルトで使用する公開鍵と秘密鍵
<ul style="list-style-type: none"><li>• plugin.conf</li><li>• plugin/</li></ul>	プラグインの設定ファイルとディレクトリ (プラグイン使用時のみ使用)

今回は**cefnetd.conf**・**cefnetd.fib**・**csmgrd.conf**を設定

\*1: 環境変数\$CEFORE\_DIRを変更してインストールした場合は"\$CEFORE\_DIR/cefore"下に存在

## (2-1) cefnetd.conf の設定

### ■ 設定ファイル cefnetd.conf の内容

```
cefore:~/cefore$ cat /usr/local/cefore/cefnetd.conf
```

```
#  
# cefnetd.conf  
#  
  
#  
# Port number used by cefnetd.  
# This value must be higher than 1024 and lower than 65536.  
#  
#PORT_NUM=9896  
#  
# Socket ID used by cefnetd.  
# This value is the string type, not the integer type.  
#  
#LOCAL SOCK_ID=0  
...
```

"#" で始まる行はコメント行

インストール直後の雛形ではすべての  
パラメータがコメントアウトされている  
(雛形のコメントに書かれている値は  
各パラメータのデフォルト値)



# cefnetd.conf の主なパラメータ

## ■ 「parameter=value」 の書式で記述する

- 例: キャッシュ無しモードからcsmgrd使用モードに変更する場合
  - CS\_MODE=2

## ■ キャッシュを使用する場合に設定すべきパラメータ

パラメータ	説明	デフォルト	値の範囲・意味
CS_MODE	CSの動作モード	0	0 : CSを使用しない 1 : cefnetdのローカルキャッシュ 2 : csmgrd
BUFFER_CAPACITY	cefnetdの最大Dataバッファサイズ	30000	$0 \leq n < 65536$
CSMGR_NODE	cefnetdが接続するcsmgrdのIPアドレス	localhost	
CSMGR_PORT	cefnetdが接続するcsmgrdのTCPポート番号	9799	$1024 < p < 65536$

# cefnetd.conf の詳細パラメータ①

## ■ cefnetd ネットワーク設定

パラメータ	説明	デフォルト	値の範囲・意味
PORT_NUM	cefnetdが使用するポート番号（単一のPC上でcefnetdを複数起動する場合等に設定）	9896	1024 < p < 65536
LOCAL SOCK_ID	UNIXドメインソケットのID文字列（単一のPC上でcefnetdを複数起動する場合等に設定）	0	
NODE_NAME	cefnetd のノード名、無指定の場合は IP アドレスを使用	-	

## ■ cefnetd のローカルキャッシュ設定

### ● CS\_MODE=1 の場合にのみ使用

パラメータ	説明	デフォルト	値の範囲・意味
LOCAL_CACHE_CAPACITY	キャッシュ容量（単位：Data数）	65535	1 < n < 8M
LOCAL_CACHE_INTERVAL	期限切れコンテンツチェック間隔（秒）	60	1 < n < 86400 (2 hours)
LOCAL_CACHE_DEFAULT_RCT	Dataのデフォルトのキャッシュ期限 (Recommended Cache Time; RCT) (Data が RCT を指定している場合はそちらを優先) (単位：ミリ秒)	600 (10分)	0 < n < 3600

# cefnetd.conf の詳細パラメータ②

## ■ cefnetd テーブルエントリ設定

パラメータ	説明	デフォルト	値の範囲・意味
PIT_SIZE	最大PITエントリ数	2048	$1 < n < 65536$
FIB_SIZE	最大FIBエントリ数	1024	$1 < n < 65536$
PIT_SIZE_APP	最大PITエントリ数 (アプリ用)	64	$1 < n < 1025$
FIB_SIZE_APP	最大FIBエントリ数 (アプリ用)	64	$1 < n < 1M$
BUFFER_CACHE_TIME	cefnetd のバッファへの保持時間 (ミリ秒)	10000 (10秒)	$0 < n$
REGULAR_INTEREST_MAX_LIFETIME	通常の Interest (Symbolic でないもの) のライフタイム (PIT エントリの生存時間) の最大値 (秒)。パケットに記載された値よりも優先される	2	$0 < n$
SYMBOLIC_INTEREST_MAX_LIFETIME	Symbolic Interest のライフタイム (PIT エントリの生存時間) の最大値 (秒)。パケットに記載された値よりも優先される	4	$0 < n$

# cefnetd.conf の詳細パラメータ③

## ■ cefnetd 転送戦略設定

パラメータ	説明	デフォルト	値の範囲・意味
INTEREST_RETRANSMISSION	<p>PIT 消失前に Interest が再送されたときの挙動。以下の二種類から選択</p> <ul style="list-style-type: none"> <li>• RFC8569: 同じ Face (PIT登録済みの Face) から来た場合は再送と見なして転送し、それ以外の場合は破棄する (集約する)</li> <li>• SUPPRESSIVE: 常に破棄する (集約する)</li> </ul>	RFC8569	左記
FORWARDING_INFO_STRATEGY	<p>FIB に複数転送先が設定されている場合の戦略。戦略は以下の二種類</p> <ul style="list-style-type: none"> <li>• 0: 任意の一つ (最初に登録されたもの) に転送</li> <li>• 1: すべてに転送</li> </ul>	0	左記
SYMBOLIC_BACKBUFFER	<p>Symbolic Interest (SMI) の挙動に関するパラメータ            ※詳細: SMI はリアルタイムストリーミング用なので、最後に転送した Data のチャンク番号を覚えておき、基本的にはそれよりチャンク番号が大きい (= 最新の) もののみ転送して、チャンク番号が小さい (= 過去の) ものの転送を抑制する。しかし、NW 環境が不安定な場合は順番が前後することもありうるので、このパラメータで指定した値だけ遡ったチャンク番号の Data の転送は許容する。確実に順番通りに来る場合は0でいいが、不安定な環境では値を大きくするとよい。</p>	100	$0 \leq n$

# cefnetd.conf の詳細パラメータ④

## ■ ログ設定

パラメータ	説明	デフォルト	値の範囲・意味
CEF_LOG_LEVEL	出力ログの詳細度（[0] Error のみ表示、[1] Warning+Error、[2] Error+Warning+Info）	0	$0 \leq n \leq 2$
CEF_DEBUG_LEVEL	デバッグ用ログの詳細度（configure時に"--enable-debug"を指定する必要がある）	0	$0 \leq n \leq 3$ （※n=3にすると パケットダンプまで 表示される）



## (2-2) cefnetd.fib の設定

### ■ 静的なFIBエントリの設定ファイル

- 書式 : `name (udp|tcp) ip_address[:port] ...`
- 設定例
  - `ccnx:/ udp 10.0.1.1`
  - `ccnx:/cinema tcp 10.0.2.1:8888 10.0.2.2:9999`
  - `ccnx:/news/today udp 10.0.3.1 10.0.3.2:8765 10.0.3.3:9876`

※経路を複数指定した場合、デフォルトでは利用可能な最初のノードにのみ転送する。cefnetd.fib の FORWARDING\_INFO\_STRATEGY の値を変更すれば、全ノードへの転送も選択できる。

### ■ 動的なFIBエントリの設定はcefrouteで行う

- 追加: `cefroute add name (udp|tcp) ip_address`
- 削除: `cefroute del name ip_address`





## (2-3) csmgrd.conf の設定

- 書式やファイルの場所はcefnetd.confと同じ
  - 「parameter=value」の形式で記述
  - "#"で始まる行はコメント
  - /usr/local/ceforeに配置

```
cefore:~/cefore$ cat /usr/local/cefore/csmgrd.conf
#
# csmgrd.conf
#
#
# Port number used by csmgrd.
# This value must be higher than 1024 and lower than 65536.
#
#PORT_NUM=9799
#
# Socket ID used by csmgrd and cefnetd.
# This value is the string type, not the integer type.
#
#LOCAL SOCK_ID=0
...
```

# NICT csmgrd.conf の主なパラメータ

パラメータ	説明	デフォルト	値の範囲・意味
CACHE_TYPE	csmgrdが使用するPlugin名称（文字列）	filesystem	<ul style="list-style-type: none"> <li>filesystem</li> <li>memory (詳細は後述)</li> </ul>
CACHE_INTERVAL	csmgrdの期限切れコンテンツチェック間隔 (単位：ミリ秒)	10,000 (10秒毎)	$1,000 < n < 86,400,000$ (1秒～24時間)
CACHE_DEFAULT_RCT	Dataのデフォルトのキャッシュ期限 (Recommended Cache Time; RCT) (Data が RCT を指定している場合はそちらを優先) (単位：ミリ秒)	600,000 (10分間)	$1,000 < n < 3,600,000$ (1秒～24時間)
CACHE_ALGORITHM	キャッシュ置換アルゴリズムライブラリ	libcsmgrd_lru	<ul style="list-style-type: none"> <li>None</li> <li>libcsmgrd_lru</li> <li>libcsmgrd_lfu</li> <li>libcsmgrd_fifo</li> </ul>
CACHE_PATH	ファイルシステムキャッシュのキャッシュ保存用ディレクトリ（ファイルシステムキャッシュ使用時は必須）	/usr/local/cefore	
CACHE_CAPACITY	キャッシュ容量（単位：Data数）	819,200	$1 < n < 64 \text{ G}$

# csmgrd.conf の詳細パラメータ①

パラメータ	説明	デフォルト	値の範囲
PORT_NUM	csmgrdが使用するポート番号	9799	1024 < p < 65536
ALLOW_NODE	<ul style="list-style-type: none"> <li>csmgrdへの接続を許可するホストのIPアドレス</li> <li>リモートでのcsmgrdへの接続を許可する場合のみ設定（デフォルトではローカルホストのみ接続可能）</li> <li>"ALL"と記述すると、全ての接続を許可</li> <li>「,（カンマ）」区切りで複数指定可能</li> <li>複数行に分けての指定も可能</li> <li>サブネットを使用した指定も可能</li> <li>設定例 <ul style="list-style-type: none"> <li>ALLOW_NODE=192.168.1.1,192.168.1.2</li> <li>ALLOW_NODE=192.168.2.0/24</li> </ul> </li> </ul>	localhost	
CEF_LOG_LEVEL	出力ログの詳細度（[0] Error のみ表示、 [1] Warning+Error、 [2] Error+Warning+Info）	0	$0 \leq n \leq 2$
CEF_DEBUG_LEVEL	デバッグ用ログの詳細度（configure時に"--enable-debug"を指定する必要有）	0	$0 \leq n \leq 3$
LOCAL SOCK_ID	UNIXドメインソケットのID文字列（単一のPC上でcsmgrdを複数起動する場合等に設定）	0	

# csmgrd.conf の詳細パラメータ②

パラメータ	説明	デフォルト	値の範囲
CACHE_ALGO_NAME_SIZE	想定される Data の平均ネーム長（単位：Byte） （キャッシュの使用メモリ領域の計算に使用）	256	$100 < n < 8000$
CACHE_ALGO_COB_SIZE	想定される Data の平均パケットサイズ（単位：Byte） （キャッシュの使用メモリ領域の計算に使用）	2048	$500 < n < 64\text{ K}$

## ③設定ファイルの変更

- (3-1) ルーティングテーブルの設定
  - **cefnetd.fibを変更**してFIBエントリを追加
  - cefstatusでFIBエントリの更新を確認
- (3-2) cefnetdがcsmgrdを利用するように設定
  - **cefnetd.confで"CS\_MODE=2"**に設定
  - **csmgrd.conf**でキャッシュの挙動を設定
  - csmgrdstart・cefnetdstartの順で起動後、動作確認

# (3-1) ルーティングテーブルの設定

## ■ cefnetd.fibに以下を入力

```
ccnx:/hoge udp 10.0.0.1
```

## ■ cefnetdを起動し、FIBエントリを確認

```
$ cefnetdstart
```

```
$ cefstatus
```

```
Version      : f0
Port         : 9896
Rx Frames    : 0
Tx Frames    : 0
Cache Mode   : None
Faces :
  faceid =    4 : IPv4 Listen face (udp)
  faceid =    0 : Local face
  faceid =   17 : Local face
  faceid =    5 : IPv6 Listen face (udp)
  faceid =   16 : address = 10.0.0.1:9896 (udp)
  faceid =    6 : IPv4 Listen face (tcp)
  faceid =   18 : IPv6 Listen face (tcp)
```

FIBエントリが  
追加されている

```
FIB :
  ccnx:/hoge/
  Faces : 16 (-s-)
```

Entry is empty

## (3-2) cefnetd のキャッシュ利用設定①

### ■ cefnetd.confで"CS\_MODE=2"に設定

cefnetd.conf

```
...  
## Content Store used by cefnetd  
# 0 : No Content Store  
# 1 : Use cefnetd's Local cache  
# 2 : Use external Content Store (use csmgrd)  
#CS_MODE=0  
CS_MODE=2 ← 追加  
...
```

### ■ cefnetd上のローカルキャッシュを使う場合は "CS\_MODE=1"に設定

# (3-2) cefnetd のキャッシュ利用設定②

## ■ csmgrd.confでキャッシュの挙動を設定（任意）

csmgrd.conf

```

...
#
# Type of CS space used by csmgrd.
# filesystem : UNIX filesystem
# memory      : Memory
#

```

- filesystem:
  - ファイルにキャッシュデータを保存
  - CACHE\_PATHパラメータでキャッシュディレクトリを変更可
- memory:
  - メモリ上にキャッシュデータを保存

```

#CACHE_TYPE=filesystem

```

CACHE\_TYPE=memory

追加

```

#
# Type of cache policy by cache plugin.
#

```

```

#CACHE_ALGORITHM=libcsmgrd_lru

```

CACHE\_ALGORITHM=libcsmgrd\_fifo

追加

- libcsmgrd\_fifo: First-In First-Out
- libcsmgrd\_lru: Least Recently Used
- libcsmgrd\_lfu: Least Frequently Used
- None: 置換を行わない  
(一杯になったらキャッシュを停止)



## (3-2) cefnetd のキャッシュ利用設定③

- csmgrdstart ・ cefnetdstart の順で起動後、動作確認
  - **cefnetd から csmgrd へ接続を行うため、最初に csmgrd を起動する**

```
$ sudo csmgrdstart          # 任意のディレクトリにアーカイブを解凍
$ sudo cefnetdstart
$ csmgrstatus ccnx:/        # この時点ではキャッシュが無い
$ echo hello > test
$ cefputfile ccnx:/test     # ファイルtestを作成してアップロード
$ csmgrstatus ccnx:/        # ccnx:/testがキャッシュされたのを確認
```

# csmgrstatus を用いたキャッシュ確認

## キャッシュが無い場合

```
cefore:~/cefore$ csmgrstatus ccnx:/

Connect to 127.0.0.1:9799
2018-08-24 13:47:56.844 [csmgrd] INFO: Open TCP peer: 127.0.0.1:37958, socket : 6
***** Connection Status Report *****
All Connection Num      : 1

***** Cache Status Report *****
Number of Cached Contents : 0
```

## キャッシュ(ccnx:/test)が有る場合

```
cefore:~/cefore$ csmgrstatus ccnx:/

Connect to 127.0.0.1:9799
2018-08-24 13:48:13.518 [csmgrd] INFO: Open TCP peer: 127.0.0.1:37960, socket : 6
***** Connection Status Report *****
All Connection Num      : 1

***** Cache Status Report *****
Number of Cached Contents : 1
```

```
[0]
Content Name : ccnx:/test/
Content Size : 6 Bytes
Access Count : 0
Freshness    : 293 Sec
Elapsed Time : 6 Sec
```

キャッシュ済みコンテンツの  
情報が表示される

```
2018-08-24 13:48:13.519 [csmgrd] INFO: Close TCP peer: 127.0.0.1:37960
```

## ③ cefputfileでファイルアップロード

1. 簡単な自己紹介文を書いたintro.txtを作成する

- 例：「私は[所属]の[名前]です。」

2. コンテンツには以下の規則で名前をつける

- 例：無線LAN icntest01に接続するaliceさんの場合

```
ccnx:/icntest01/alice/[ファイル名]
```

無線LANのSSID

自身の名前（名簿のPublisher識別子）

3. intro.txtをアップロードする

```
$ cefputfile ccnx:/icntest01/alice/intro.txt -e 7200 -t 7200
```

上記規則に従い自分のコンテンツ名を決定

キャッシュ期限オプション（2時間）



# cefputfileの動作例

```
cefore:~/cefore$ cefputfile ccnx:/icntest01/alice/intro.txt -e 7200 -t 7200
```

```
[cefputfile] Start
[cefputfile] Parsing parameters ... OK
[cefputfile] Init Cefore Client package ... OK
[cefputfile] Conversion from URI into Name ... OK
[cefputfile] Checking the input file ... OK
[cefputfile] Connect to cefnetd ... OK
[cefputfile] URI           = ccnx:/icntest01/alice/intro.txt
[cefputfile] File          = intro.txt
[cefputfile] Rate          = 5.000 Mbps
[cefputfile] Block Size    = 1024 Bytes
[cefputfile] Cache Time    = 7200 sec
[cefputfile] Expiration    = 7200 sec
[cefputfile] Start creating Content Objects
[cefputfile] Unconnect to cefnetd ... OK
[cefputfile] Terminate
[cefputfile] Tx Frames = 1
[cefputfile] Tx Bytes  = 6
[cefgetfile] Duration  = 0.004 sec
[cefputfile] Thoroughput = 18140 bps
```

```
cefore:~/cefore$ csmgrstatus ccnx:/
```

```
Connect to 127.0.0.1:9799
```

```
...
Content Name : ccnx:/icntest01/alice/intro.txt,
Content Size : 6 Bytes
Access Count : 0
Freshness    : 7193 Sec
Elapsed Time : 4 Sec
```

ファイルがキャッシュされている  
(キャッシュされていない場合は  
cefnetd.confでUSE\_CACHE=1に  
なっているかどうかを確認)

## ④ cefgetfileでファイルダウンロード

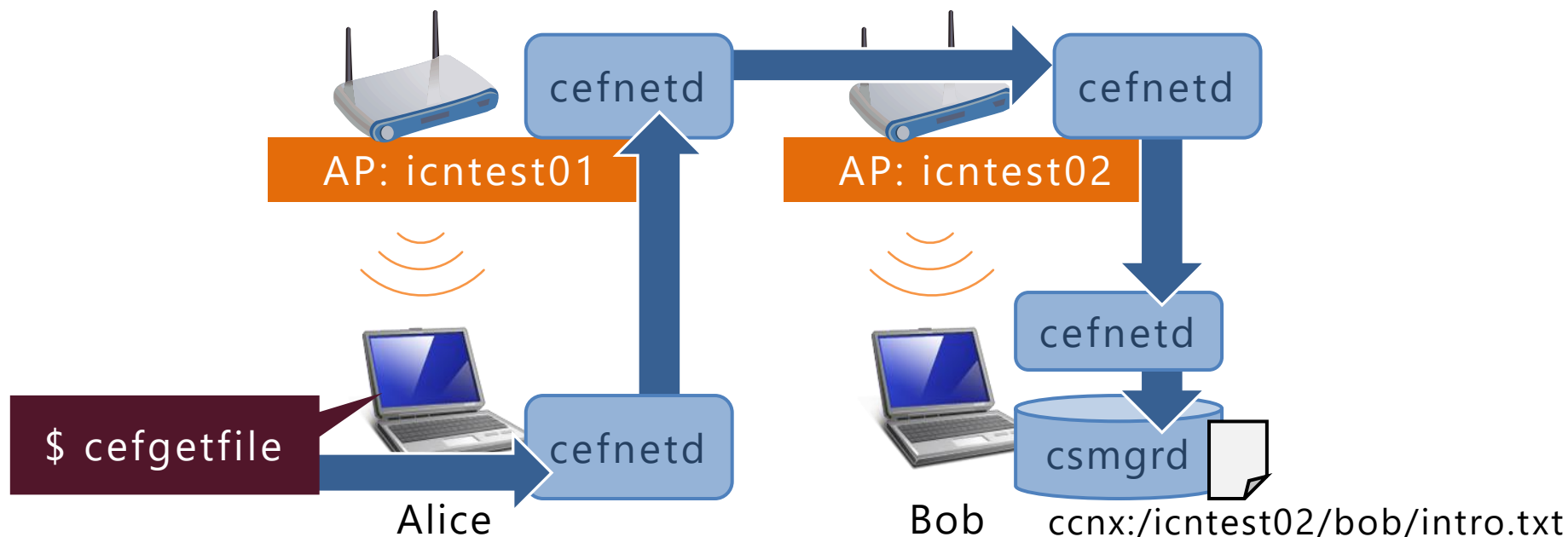
### ■ 名簿を参考に他の人のintro.txtをダウンロードする

- 例：icntest02のBobのintro.txtをダウンロードして  
bob.txtとして保存する

```
$ cefgetfile ccnx:/icntest02/bob/intro.txt -f bob.txt
```

```
$ cat bob.txt
```

保存先ファイル名をオプションで指定





# cefgetfileの動作例

```
cefore:~/cefore$ cefgetfile ccnx:/icntest02/bob/intro.txt -f bob.txt
```

```
[cefgetfile] Start
[cefgetfile] Parsing parameters ... OK
[cefgetfile] Init Cefore Client package ... OK
[cefgetfile] Conversion from URI into Name ... OK
[cefgetfile] Checking the output file ... OK
[cefgetfile] Connect to cefnetd ... OK
[cefgetfile] URI=ccnx:/icntest02/bob/intro.txt
[cefgetfile] Start sending Interests
[cefgetfile] Complete
[cefgetfile] Unconnect to cefnetd ... OK
[cefgetfile] Terminate
[cefgetfile] Rx Frames = 1
[cefgetfile] Rx Bytes = 26
[cefgetfile] Duration = 0.000 sec
[cefgetfile] Jitter (Ave) = 0 us
[cefgetfile] Jitter (Max) = 0 us
[cefgetfile] Jitter (Var) = 0 us
cefore:~/cefore$ cat bob.txt
私はNICTのBobです。
```

コンテンツをダウンロードして  
bob.txtに保存できている



# その他のツールの使用

- cefgetfile/cefputfileのオプションを確認しよう
  - ユーザマニュアル6.1節、6.2節
  - 大きなサイズのファイルを作ってアップロード速度やダウンロード速度を計測してみよう
    - ・ cefputfileはアップロードレートが調整可能(rオプション)
    - ・ cefgetfileは取得パイプライン数が調整可能(sオプション)
- cefgetchunkで複数のチャンクから成るコンテンツの特定のチャンクだけ取得してみよう
  - ユーザマニュアル6.3節
- cefinfoで遅延などのネットワーク情報を観測してみよう
  - ユーザマニュアル6.6節、6.7節
  - --enable-cefping, --enable-cefinfoのつけ忘れに注意

## 第16回ICN研究会ワークショップ

# Ceforeアプリ開発ツール cefpyco

---

2021年 8月26日（木）・ 8月27日（金）



- cefnetdへの接続
- Dataパケットの送信\*1
- Interestパケットの送信
- パケットの受信
- 簡易Consumerアプリの作成
- 簡易Producerアプリの作成
- サンプルアプリCefApp

\*1: 本スライドではCob(Content Object)をDataまたはDataパケットと呼称

# cefpyco を用いた通信

---



# cefpycoを用いた通信

- ① cefnetdへの接続
  - create\_handle()メソッド
- ② Dataパケットの送信
  - send\_data(name, payload , chunk\_num)メソッド
- ③ Interestパケットの送信
  - send\_interest(name, chunk\_num)メソッド
- ④ パケットの受信
  - receive()メソッド
- ⑤ 簡易Consumerアプリの作成
- ⑥ 簡易Producerアプリの作成
  - register(name)メソッド

# ① cefnetdへの接続

1. 以下の内容のpythonファイルtest1.pyを作成

```
import cefpyco

with cefpyco.create_handle() as handle:
    pass # ブロック開始時にcefnetdへ接続、終了時に切断
```

test1.py

2. cefnetdを起動して実行（エラーが無ければ正常）

```
cefore:~/cefpyco$ sudo csmgrdstart # if needed
cefore:~/cefpyco$ sudo cefnetdstart
cefore:~/cefpyco$ sudo python test1.py
2021-08-26 15:14:00.123 [cefpyco] INFO: [client]
Config directory is /usr/local/cefore
2021-08-26 15:14:00.123 [cefpyco] INFO: [client]
Local Socket Name is /tmp/cef_9896.0
2021-08-26 15:14:00.123 [cefpyco] INFO: [client]
Listen Port is 9896
```



# 補足：pythonの文法

## ■ C言語等のセミコロンや括弧の代わりにインデントで文・ブロックを表現

ブロックの範囲を  
一目で見分けられる

```
# a=1, b=1のときはbと表示
# a=1, b≠1のときはaと表示
# a≠1のときは何もしない
if a == 1:
    if b == 1:
        print("b")
    else:
        print("a")
```

インデント幅が揃っていないと  
バグ扱いなので要注意

```
if a == 1:
    print("correct")
else:
    print("error")
    print("error")
```

Tab文字

エラー例

- 空白4文字と空白2文字
- 空白4文字とタブ1文字

## ■ with構文：煩雑な開始/終了/例外処理を省略できる文法

### ● 代表例：ファイルオープン・クローズ

with構文無しの場合

```
print("Begin.")
try:
    h = cefpyco.CefpycoHandle()
    h.begin()
    print("Do something.")
except Exception as e:
    print(e)
    # 例外処理
finally:
    h.end()
print("End.")
```

with構文を用いた場合

```
print("Begin.")
with cefpyco.create_handle() as h:
    print("Do something.")
print("End.")
```

## ②Dataパケットの送信

1. 以下の内容のpythonファイルtest2.pyを作成

```
import cefpyco

with cefpyco.create_handle() as handle:
    # ccnx:/testというコンテンツ名・チャンク番号0で
    # helloというテキストコンテンツをDataパケットとして送信
    handle.send_data("ccnx:/test", "hello", 0, cache_time=7200000)
```

test2.py

2. csmgrd・cefnetdを起動して実行

```
cefore:~/cefpyco$ sudo csmgrdstart
cefore:~/cefpyco$ sudo cefnetdstart
cefore:~/cefpyco$ sudo python test2.py
```

```
...
cefore:~/cefpyco$ csmgrstatus ccnx:/
```

```
...
*****      Cache Status Report      *****
Number of Cached Contents      : 1
[0]
  Content Name : ccnx:/test/
  Content Size : 5 Bytes
  Access Count : 0
...
```

csmgrdに  
Dataが  
アップロード  
される

## ③Interestパケットの送信

1. 以下の内容のpythonファイルtest3.pyを作成

```
import cefpyco
```

test3.py

```
with cefpyco.create_handle() as handle:
    # ccnx:/testというコンテンツの0番目のチャンクを
    # 要求するInterestパケットを送信
    handle.send_interest("ccnx:/test", 0)
```

2. (test2.py実行後に) test3.pyを実行

```
# (test2.pyのシナリオを実行)
```

```
cefore:~/cefpyco$ sudo python test3.py
```

```
...
```

```
cefore:~/cefpyco$ csmgrstatus ccnx:/
```

```
...
```

```
*****      Cache Status Report      *****
```

```
Number of Cached Contents      : 1
```

```
[0]
```

```
Content Name : ccnx:/test/
```

```
Content Size : 5 Bytes
```

```
Access Count
```

```
1
```

Access Count  
が 1 だけ増加

```
...
```



## ④ パケットの受信

### ■ 以下のコードでパケットを受信可能

```
import cefpyco

with cefpyco.create_handle() as handle:
    handle.send_interest("ccnx:/test", 0) # Dataパケットを受信したい場合
    # handle.register("ccnx:/test") # Interestパケットを受信したい場合
    info = handle.receive()
```

### ■ receive()メソッド

- 実行後、約4秒の間 パケットを待ち受ける
  - ・ 成功するまで受信したい場合はwhileループ等が必要
- CcnPacketInfoオブジェクトを返す
  - ・ 受信の成否やInterest/Dataに依らない
  - ・ プロパティ一覧は次ページで説明





# CcnPacketInfoのプロパティ

プロパティ名	型	説明
<b>is_succeeded, is_failed</b>	bool	パケット受信の成否フラグ
<b>is_interest, is_data</b>	bool	受信したパケットがInterest/Dataか否かを表すフラグ (受信失敗時には両方ともFalseとなる)
<b>name</b>	string	URI形式(ccnx:/~)の名前
<b>name_len</b>	int	URI形式の名前の長さ(name TLV長ではない)
<b>chunk_num</b>	int	チャンク番号
<b>payload</b>	bytes	(Dataパケットの場合) コンテンツのデータ
<b>payload_s</b>	string	(Dataパケットの場合) コンテンツのデータ (文字列として取得、バイナリデータの場合は無効)
<b>payload_len</b>	int	(Dataパケットの場合) コンテンツのデータのバイト長
<b>version</b>	int	受信パケットのバージョン値
<b>type</b>	int	受信パケットのタイプ値
<b>actual_data_len</b>	int	受信したパケットのヘッダを含むバイト長
<b>end_chunk_num</b>	int	コンテンツの最後のチャンク番号 (指定時のみ有効)

## ⑤ 簡易Consumerアプリの作成

### ■ 以下の内容のpythonファイルconsumer.pyを作成

- チャンクが 1 個しかないコンテンツccnx:/testを受信
- 受信に成功するまでreceiveメソッドを繰り返す

### ■ 穴埋め問題：

- 受信したCcnPacketInfoを見て受信に成功したか否かを検証するif文を完成させよう

```
from time import sleep
import cefpyco
```

consumer.py

```
with cefpyco.create_handle() as handle:
```

```
    while True:
```

```
        handle.send_interest("ccnx:/test", 0)
```

```
        info = handle.receive()
```

```
        if info.is_ and (info.name == ) and (info.chunk_num == 0):
```

```
            print("Success")
```

```
            print(info)
```

```
            break
```

```
        sleep(1)
```

## ⑤簡易Consumerアプリの作成：解答

- 以下の内容のpythonファイルconsumer.pyを作成
  - チャンクが 1 個しかないコンテンツccnx:/testを受信
  - 受信に成功するまでreceiveメソッドを繰り返す
- 穴埋め問題：
  - 受信したCcnPacketInfoを見て受信に成功したか否かを検証するif文を完成させよう


```
from time import sleep
import cefpyco

with cefpyco.create_handle() as handle:
    while True:
        handle.send_interest("ccnx:/test", 0)
        info = handle.receive()
        if info.is_succeeded and (info.name == "ccnx:/test") and (info.chunk_num == 0):
            print("Success")
            print(info)
            break
        sleep(1)
```

consumer.py

# consumer.pyの動作例

```
cefore:~/cefpyco$ sudo csmgrdstart
...
cefore:~/cefpyco$ sudo cefnetdstart
...
cefore:~/cefpyco$ echo hello > test; cefputfile ccnx:/test
...
cefore:~/cefpyco$ sudo python consumer.py
[cefpyco] Configure directory is /usr/local/cefore
2021-08-26 15:16:00.123 [cefpyco] INFO: [client] Config directory is /usr/local/cefore
2021-08-26 15:16:00.123 [cefpyco] INFO: [client] Local Socket Name is /tmp/cef_9896.0
2021-08-26 15:16:00.123 [cefpyco] INFO: [client] Listen Port is 9896
2021-08-26 15:16:00.123 [cefpyco] INFO: Send interest (name: ccnx:/test, #chunk: 0)
Success
Info: Succeeded to receive Data packet with name 'ccnx:/test' (#chunk: 0) and payload
'hello' (6 Bytes)
```



ccnx:/testという名前の  
コンテンツの0番目のチャンクの  
取得に成功



# トラブルシューティング

## ■ ProducerがInterestを受け取れない

- cefnetdのバッファやcsmgrdのキャッシュにヒットしている可能性がある
  - /usr/local/cefore/cefnetd.confで"CS\_MODE=0"としてcsmgrdを停止する
  - cefnetdの場合はしばらく待つ

## ⑥簡易Producerアプリの作成

■ 以下の内容のpythonファイルproducer.pyを作成

● **Interestを受信するにはregisterメソッドを用いる**

- register(prefix): cefnetdがInterestを受信した際に、Interestの名前とprefixが一致する場合はアプリに送るように登録するためのメソッド

■ 穴埋め問題：

- "ccnx:/test"宛のInterestを受け取ったら"hello"という文字列を返すifブロックを完成させよう

```
import cefpyco
```

producer.py

```
with cefpyco.create_handle() as handle:
```

```
    handle.register("ccnx:/test")
```

```
    while True:
```

```
        info = [redacted]
```

```
        if info.is [redacted] and (info.name [redacted]) and (info. [redacted]):
```

```
            handle.send_ [redacted]
```

```
            # break # Uncomment if publisher provides content once
```

## ⑥簡易Producerアプリの作成：解答

■ 以下の内容のpythonファイルproducer.pyを作成

● **Interestを受信するにはregisterメソッドを用いる**

- register(prefix): cefnetdがInterestを受信した際に、Interestの名前とprefixが一致する場合はアプリに送るように登録するためのメソッド

■ 穴埋め問題：

- "ccnx:/test"宛のInterestを受け取ったら"hello"という文字列を返すifブロックを完成させよう

```
import cefpyco

with cefpyco.create_handle() as handle:
    handle.register("ccnx:/test")
    while True:
        info = handle.receive()
        if info.is_succeeded and (info.name == "ccnx:/test") and (info.chunk_num == 0):
            handle.send_data("ccnx:/test", "hello", 0, cache_time=7200000)
            # break # Uncomment if publisher provides content once
```

producer.py

# producer.pyの動作例

■ csmgrdを無効にしてから、2つの端末で以下を実行

```
cefore:~/cefpyco$ sudo cefnetdstart
```

端末 1 (producer)

...

```
cefore:~/cefpyco$ sudo python producer.py
```

```
[cefpyco] Configure directory is /usr/local/cefore
```

```
2021-08-26 15:19:00.123 [cefpyco] INFO: [client] Config directory is /usr/local/cefore
```

```
2021-08-26 15:19:00.123 [cefpyco] INFO: [client] Local Socket Name is /tmp/cef_9896.0
```

```
2021-08-26 15:19:00.123 [cefpyco] INFO: [client] Listen Port is 9896
```

コンテンツ配布開始

```
cefore:~/cefpyco$ sudo python consumer.py
```

端末 2 (consumer)

```
[cefpyco] Configure directory is /usr/local/cefore
```

```
2021-08-26 15:19:10.123 [cefpyco] INFO: [client] Config directory is /usr/local/cefore
```

```
2021-08-26 15:19:10.123 [cefpyco] INFO: [client] Local Socket Name is /tmp/cef_9896.0
```

```
2021-08-26 15:19:10.123 [cefpyco] INFO: [client] Listen Port is 9896
```

```
2021-08-26 15:19:10.123 [cefpyco] INFO: Send interest (name: ccnx:/test, #chunk: 0)
```

```
Success
```

```
Info: Succeeded in receiving Data packet with name 'ccnx:/test' (#chunk: 0) and payload  
'hello' (5 Bytes)
```

csmgrd無しで取得に成功



# CefpycoHandle API

CefpycoHandle メソッド名	引数・返り値 (key=valueはデフォルト値のある省略可能引数)	説明
begin	<ul style="list-style-type: none"> <li>• ceforedir=None: cefnetd.confの入ったディレクトリパス</li> <li>• portnum=9896: cefnetd のポート番号</li> </ul>	cefnetdへの接続を開始する。with構文を利用する場合は自動で呼ばれる。
end	無し	cefnetdへの接続を終了する。with構文を利用する場合は自動で呼ばれる。
send_interest	<ul style="list-style-type: none"> <li>• name: コンテンツ名 (ccnx:/...)</li> <li>• chunk_num=0: チャンク番号 (負の数の場合はチャンク番号無し)</li> <li>• symbolic_f=INTEREST_TYPE_REGULAR: Interestのタイプを指定</li> <li>• hop_limit=32: ホップ数</li> <li>• lifetime=4000: Interest ライフタイム (現在の時刻からのミリ秒)</li> </ul>	指定した名前のコンテンツを要求する Interest パケットを生成して送信する。
send_data	<ul style="list-style-type: none"> <li>• name: コンテンツ名 (ccnx:/...)</li> <li>• payload: Dataパケットのペイロード</li> <li>• chunk_num=-1: チャンク番号 (負の数の場合はチャンク番号無し)</li> <li>• end_chunk_num=-1: コンテンツの最後のチャンク番号 (負の数の場合は省略)</li> <li>• hop_limit=32: 最大ホップ数</li> <li>• expiry=36000000: コンテンツ期限 (現在の時刻からのミリ秒)</li> <li>• cache_time=-1: 推奨キャッシュ時間 (負の数の場合は省略)</li> </ul>	指定した名前とペイロードに基づいて Data パケットを生成して送信する。
receive	<ul style="list-style-type: none"> <li>• error_on_timeout=false: タイムアウト時にエラーを投げるか否か</li> <li>• timeout_ms=4000: 受信開始からタイムアウトまでの時間 (ミリ秒)</li> <li>• 返り値: CcnPacketInfo (別スライド参照)</li> </ul>	InterestまたはDataパケットを指定した時間だけ待ち受け (デフォルト4秒)、受信パケットの情報を返す。
register	<ul style="list-style-type: none"> <li>• name: 受信したいInterestのプレフィックス名を指定</li> </ul>	受信したい Interest のプレフィックス名を cefnetd に登録し、receive で Interest を受け取れるようにする。
deregister	<ul style="list-style-type: none"> <li>• name: 登録を解除したい名前を指定</li> </ul>	register で登録した名前を解除する。
send_symbolic_interest	chunk_numとsymbolic_fが無い以外はsend_interestと同様	通常の Interest の代わりに Symbolic Interest (Cefore 独自機能) を送信する。



# サンプルアプリ CefApp

## ■ cefpyco/cefappディレクトリ内の2つのアプリ

- cefappconsumer.py: Consumerアプリ
- cefappproducer.py: Producerアプリ

## ■ 特徴

- 先ほどの簡易アプリと異なり、複数チャンクから成るコンテンツの送受信に対応
- 入出力はインライン・標準入出力・ファイルの3種類
- cefappconsumerはパイプライン処理を実装

## ■ 詳細な使用法はREADME 5章参照



# CefApp動作例

```
cefore:~/cefpyco$ sudo cefnetdstart
```

端末 1 (producer)

...

```
cefore:~/cefpyco$ ./cefappproducer.py ccnx:/test hello
```

```
[cefpyco] Configure directory is /usr/local/cefore
```

```
YYYY-MM-DD hh:mm:ss.xxx [cefpyco] INFO: [client] Config directory is /usr/local/cefore
```

```
YYYY-MM-DD hh:mm:ss.xxx [cefpyco] INFO: [client] Local Socket Name is /tmp/cef_9896.0
```

```
YYYY-MM-DD hh:mm:ss.xxx [cefpyco] INFO: [client] Listen Port is 9896
```

```
[cefapp] Receiving Interest...
```

コンテンツ配布開始

```
cefore:~/cefpyco$ sudo ./cefappconsumer.py ccnx:/test
```

端末 2 (consumer)

```
[cefpyco] Configure directory is /usr/local/cefore
```

```
YYYY-MM-DD hh:mm:ss.xxx [cefpyco] INFO: [client] Config directory is /usr/local/cefore
```

```
YYYY-MM-DD hh:mm:ss.xxx [cefpyco] INFO: [client] Local Socket Name is /tmp/cef_9896.0
```

```
YYYY-MM-DD hh:mm:ss.xxx [cefpyco] INFO: [client] Listen Port is 9896
```

```
YYYY-MM-DD hh:mm:ss.xxx [cefpyco] INFO: Send interest (name: ccnx:/test/meta, #chunk: 0)
```

```
YYYY-MM-DD hh:mm:ss.xxx [cefpyco] INFO: Send interest (name: ccnx:/test, #chunk: 0)
```

```
[cefapp] Succeed to receive.
```

```
hello
```

コンテンツ取得に成功

- 複数チャンクの送受信を試してみよう
  - send\_interestやsend\_dataをチャンク数に応じて複数回実行するしてみよう
  - end\_chunk\_numを指定して取得すべきチャンク数だけ要求を出すようにしてみよう
- cefgetfile/cefputfileと同等の機能を作ってみよう
  - ファイルを読み書きして送受信できるようにしよう
  - 大きなファイルに対応するためのパイプライン処理を作ってみよう
    - CefAppのコード等を参考
- cefpyco/c\_src/cefpyco.cやceforeのツールを参考に、C言語でアプリを作ってみよう





# Cefore のインストール

## ■ 自身で環境構築を行う場合の手動インストール手順

- ① ソースコードとマニュアルのダウンロード
- ② ビルドとインストール
- ③ インストールされる機能について
  - ・ デフォルトでインストールされる機能
  - ・ インストール時にオプション指定が必要な機能
- ④ バッファチューニング

# ①ダウンロード (cefore.net の場合)

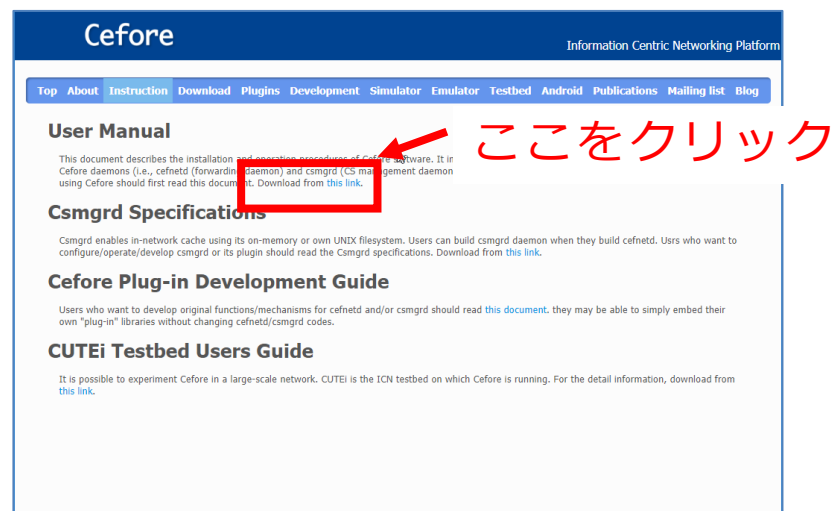
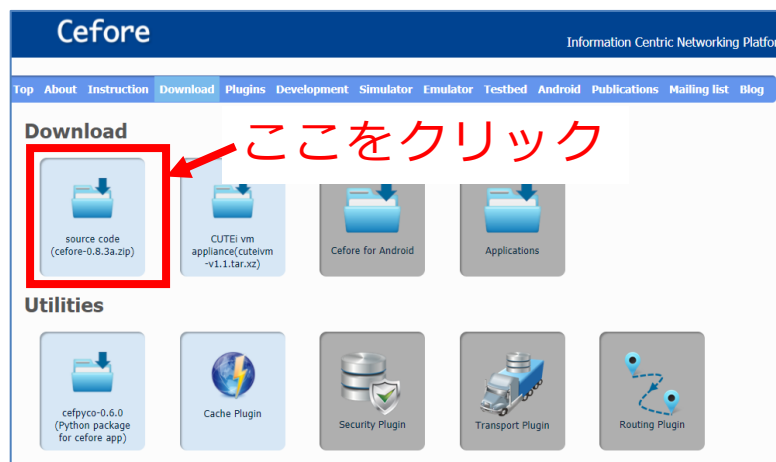
<https://cefore.net/>から  
ソースコードとユーザマニュアルをダウンロード

## ■ ソースコード

- Download
    - > source code
- (cefore-0.8.3a.zip)

## ■ ユーザマニュアル

- Instruction
  - > User Manual



# ①ダウンロード (github.com の場合)

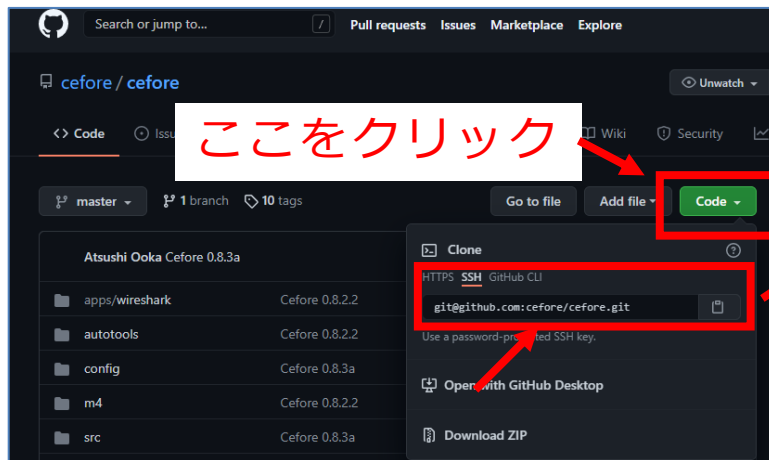
<https://github.com/cefore/cefore> から  
ソースコードをクローン

## ■ ソースコード

### ● Code

> HTTPS または SSH の  
アドレスをコピー

## ■ 自身の環境で git clone



```
$ git clone git@github.com:cefore/cefore.git
```

ここをコピー





## ②ビルドとインストール(Ubuntu)

### ■ライブラリのインストール

```
$ sudo apt-get install libssl-dev automake
```

### ■Ceforeのビルド

```
$ unzip cefore-0.8.3a.zip # 任意のディレクトリでアーカイブを解凍
$ cd cefore-0.8.3a
$ autoconf
$ automake
$ ./configure --enable-csmgr --enable-cache # csmgr、ローカルキャッシュを有効化
$ make
$ sudo make install # /usr/local/bin, sbin にインストールされる
$ sudo ldconfig # (必要に応じて実行)
```



## ② ビルドとインストール(MacOS)

### ■ ライブラリのインストール (homebrew使用時)

```
$ brew install openssl automake
```

### ■ Ceforeのビルド

```
$ unzip cefore-0.8.3a.zip # 任意のディレクトリにアーカイブを解凍
```

```
$ cd cefore-0.8.3a
```

```
$ autoconf
```

```
$ automake
```

```
$ export PATH="/usr/local/sbin:/usr/local/opt/openssl/bin:$PATH"
```

```
$ ./configure --enable-csmgr --enable-cache  
openssl_header_path=/usr/local/opt/openssl/include/  
LDFLAGS='-L/usr/local/opt/openssl/lib'  
CPPFLAGS='-I/usr/local/opt/openssl/include/'
```

```
$ make
```

```
$ sudo make install
```

一行で入力して実行  
※長いので打ち間違いに注意  
(macportsの場合は  
"/usr/local/opt/openssl"を  
"/opt/local"に置き換える)

### ■ ~/.bash\_profileに以下を追加

```
export PATH="/usr/local/sbin:/usr/local/opt/openssl/bin:$PATH"
```



# トラブルシューティング: make

## ■ autoconfに失敗する

- →aclocalを実行する
  - brewでaclocalが失敗する場合は"brew doctor; brew brune"を試す

## ■ Mac

- cefctrlやcsmgrdが見つからないというエラーが出る
  - →PATHに/usr/local/sbinが入っているか確認する
- configureに失敗する
  - →オプションを打ち間違いしていないか確認する
    - × openssl\_header\_path ○opssl\_header\_path
    - × LDFLAGS=`-L...` (バッククオート)
    - ○ LDFLAGS=' -L...' (シングルクオート)
  - configureのオプションを.bash\_profileに追加すれば毎回の入力を省略可能
    - .bash\_profile に以下を書き込む (要端末再起動)

```
$ export PATH="/usr/local/sbin:/usr/local/opt/openssl/bin/:$PATH"
$ export LDFLAGS="-L/usr/local/opt/openssl/lib"
$ export CPPFLAGS="-I/usr/local/opt/openssl/include"
$ export opssl_header_path="/usr/local/opt/openssl/include"
```

### ③ インストールされる機能（デフォルト）

機能	形態	説明
cefnetd	daemon	フォワーディングデーモン
cefnetdstart	utility	フォワーディングデーモン起動ユーティリティ
cefnetdstop	utility	フォワーディングデーモン停止ユーティリティ
cefstatus	utility	cefnetdのstatus標準出力ユーティリティ
cefroute	utility	FIB操作ユーティリティ
cefputfile	tool	任意のファイルをNamed Cobに変換しcefnetdへ入力する
cefgetfile	tool	cefnetdを介して取得したコンテンツをファイルとして出力する
cefgetchunk	tool	指定されたNamed Cobを取得し、ペイロードを標準出力する
cefputstream	tool	標準入力をNamed Cobに変換しcefnetdへ入力する
cefgetstream	tool	cefnetdを介して取得したコンテンツを標準出力する

### ③ インストールされる機能（要オプション）

#### ■ configure実行時にオプション指定が必要な機能

- 例: csmgrdとcefpingを有効化する場合
  - `./configure --enable-csmgr --enable-cefping`
- configure変更後はmakeを再実行

機能	形態	option	説明
(cefnetdローカルキャッシュ)	function	--enable-cache	cefnetdのローカルキャッシュを有効化(CS_MODE=1)
csmgrd	daemon	--enable-csmgr	コンテンツストア管理デーモン (CS_MODE=2)
csmgrdstart	utility	--enable-csmgr	csmgrd起動ユーティリティ
csmgrdstop	utility	--enable-csmgr	csmgrd停止ユーティリティ
csmgrstatus	utility	--enable-csmgr	csmgrdのstatus標準出力ユーティリティ
csmgrecho	tool	--enable-csmgr	csmgr接続確認ツール
cefping	tool	--enable-cefping	ネットワーク管理ツールcefping
Sample Transport	plugin	--enable-samtp	Sample Transport プラグイン
NDN Plugin	plugin	--enable-ndn	NDNプラグイン



# インストールディレクトリの指定方法

- 環境変数"\$CEFORE\_DIR"でインストール先を指定可能
  - "\$CEFORE\_DIR"のデフォルトは"/usr/local"
  - daemon機能は"\$CEFORE\_DIR/sbin"
  - utilityとtool機能は"\$CEFORE\_DIR/bin"
  - 設定ファイルは"\$CEFORE\_DIR/cefore"
  
- インストールディレクトリを変更した場合は、  
configure実行前にautoconfとautomakeを再実行

## ④ バッファチューニング

### ■ Linux OS

```
$ sudo sysctl -w net.core.rmem_default=10000000  
$ sudo sysctl -w net.core.wmem_default=10000000  
$ sudo sysctl -w net.core.rmem_max=10000000  
$ sudo sysctl -w net.core.wmem_max=10000000
```

### ■ Mac OS

```
$ sudo sysctl -w net.local.stream.sendspace=2000000  
$ sudo sysctl -w net.local.stream.recvspace=2000000
```

- PC再起動時にパラメータが初期化されるので、再実行しやすいようスクリプト化するのを推奨

# トラブルシューティング: csmgrd

- csmgrdstartコマンドが見つからない。
  - →configure実行時に"--enable-csmgr"を付けているか確認する。
    - ・ オプション指定にミスがあると無視されるので、打ち間違いに要注意。
- configureのオプション指定を変更すると、makeに失敗する。
  - "make clean"を実行してからmakeをやり直す。
- csmgrdstart実行時に" [csmgrd] ERROR: libcsmgrd\_plugin.so: cannot open shared object file: No such file or directory"と表示される。
  - →Ubuntuの場合、"sudo ldconfig"を実行する。
  - →MacOSの場合、以下を実行する（~/bash\_profileにも要追記）。
    - ・ export PATH="/usr/local/sbin:/usr/local/opt/openssl/bin:\$PATH"
- cefnetd・csmgrdが起動しない
  - "sudo cefnetdstop -F" を実行してから再起動する。
  - 「インストール④バッファチューニング」を行ったかどうか確認する。
    - ・ バッファチューニングはPCを再起動すると設定が初期化されるので要注意。



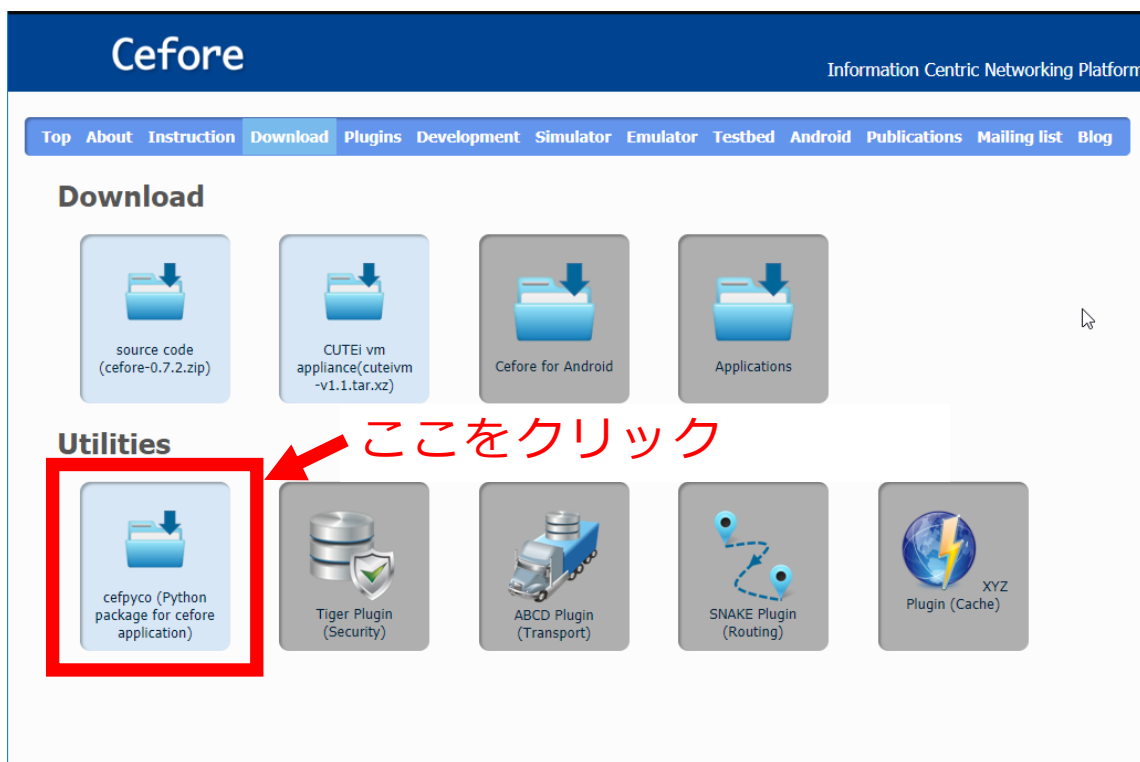
# cefpyco のインストール

---

Docker を使わず自身で環境構築する場合の導入手順

# NICT cefpycoのダウンロード

- <https://cefore.net/download> > Utilities > cefpyco
  - マニュアル (README.html) 付属\*1



\*1: 2018/8/30現在日本語マニュアルのみ



# ライブラリのインストール

## ■ ライブラリのインストール (Ubuntu)

### ● python3 環境を推奨

```
$ sudo apt-get install cmake python-pip
```

```
$ sudo pip install setuptools click numpy
```

## ■ ライブラリのインストール (Mac)

```
$ brew install cmake pyenv
```

```
$ pyenv install 3.7.3 # 任意のバージョンを指定
```

```
$ pyenv global 3.7.3 # pyenv local ... なら実行ディレクトリでのみ有効化
```

```
$ eval "$(pyenv init -)"
```

```
$ pyenv versions # インストール version の確認
```

```
$ sudo easy_install pip # macports では py-pip
```

```
$ python -m pip install setuptools click numpy
```

```
$ echo 'eval "$(pyenv init -)"' >> ~/.bash_profile # 任意
```



# ビルドとインストール

## ■ cefpycoのビルド

\$ unzip cefpyco-0.6.0.zip	# 任意のディレクトリにアーカイブを解凍
\$ cd cefpyco.release	
\$ cmake .	# "."のつけ忘れに注意
\$ sudo make install	
\$ cd test	
\$ sudo python	# pythonでインポートして起動確認
>> import cefpyco	# エラーが起きなければ正常
>> (Ctrl-D)	# Ctrl-D を押して終了

# (再掲) バッファチューニング

## ■ Linux OS

```
$ sudo sysctl -w net.core.rmem_default=10000000
```

```
$ sudo sysctl -w net.core.wmem_default=10000000
```

```
$ sudo sysctl -w net.core.rmem_max=10000000
```

```
$ sudo sysctl -w net.core.wmem_max=10000000
```

## ■ Mac OS

```
$ sudo sysctl -w net.local.stream.sendspace=2000000
```

```
$ sudo sysctl -w net.local.stream.recvspace=2000000
```