e2TP メッセージング API 仕様書

TEF950-S002-01.00.00/ja

2005 年 3 月
T-Engine フォーラム

e²TP メッセージング API 仕様書

規定範囲 (Scope)

本文書は，Java プログラムから TENeT 標準における e²TP (extended eTRON Transfer Protocol) メッセージを送受するための Java API 仕様である e²TP メッセージング API を定義する．

参照規定 (Normative References)

e²TP メッセージング API:

> e²TP メッセージ仕様書, T-Engine Forum, 2005.
> Gosling, J., Joy, B., Steele, G., and Bracha, G.: The Java Language Specification, Sun Microsystems, 2000.

VTS API:

> TENeT メッセージ仕様書, T-Engine Forum, 2005.
> Gosling, J., Joy, B., Steele, G., and Bracha, G.: The Java Language Specification, Sun Microsystems, 2000.

はじめに

e²TP メッセージング API は，TENeT で用いられる通信メッセージである e²TP メッセージを，IC カードとの間や他のアプリケーションプログラム (AP) との間で送受するための API である．

本 API を用いて，AP は e²TP メッセージを同期 (送信メッセージに対する返答メッセージをブロック待ち受けする) もしくは非同期 (メッセージの送信後，返答メッセージを待たずに処理を続行する) のいずれかにより送受信することができる (see DispatchAgent#syncSend()/send()). 非同期メッセージ送信を行った場合，受信メッセージはあらかじめ登録したリスナにより取得することとなる (see MessageListener#received()).

本 API により送信される e²TP メッセージの宛先としては， TENeT 名前空間の eTRON ID を持つ，任意の AP や IC カード，サーバなどを指定することができる．宛先がリモートであった場合には，メッセージは自動的にルーティングされ，ネットワークなどの適切な経路を経由して送信される． AP は宛先がローカルであるかリモートであるかを意識する必要はない．ただし，セキュリティ上の理由により，一部のメッセージはリモートから送信された場合は無効となる．リモート送信時に無効になるメッセージの種別については，TENeT メッセージ仕様を参照されたい．

本 API において，各 AP の eTRON ID は IC カードの eTRON ID を用いて自動的に生成お

よび取得される (see SystemManager#getAgent()). この値は DispatchAgent オブジェクトに格納され, メッセージ送信時に自動的にメッセージの送信元 ID として付与される. そのため, 各 AP は eTRON ID の具体的な割り当てについて意識する必要はない.

以下に, 本 API の利用サンプルを示す.

```
// 1. eTRON ID の取得 (メッセージ授受のためのエントリポイント取得)

 String domain =
  (String) SystemManager.getInstance().getDomainMap().get(IccRwName);
 // ドメインの取得. IccRwName は PC/SC における R/W 名など環境内で IC
 // カードを一意に識別するための名前とする.

 DispatchAgent agent =
  SystemManager.getInstance().getAgent(domain);
 // メッセージ送受のためのエントリポイント取得. ここで eTRON ID は AP
 // に対して自動的に払い出される. 以後、メッセージ送信の際に送信元 ID
 // として、ここで払い出された ID が自動的に使われる (AP が送信元 ID
 // を詐称することはできない).

// 2. メッセージの送受信

 ETronID dst = new ETronID(domain, 0);
 // IC カードの eTRON ID はポート 0.

 int createFile = 0x0040;
 // CreateFile メッセージ (TENeT メッセージ仕様書参照).

 Message msg = new Message(dst, createFile, fileData);
 // 送信するメッセージの作成. fileDataは作成する権利価値のデータとする.

 Message reply = agent.syncSend(msg);
 // 同期メッセージ送受信. メッセージ msg を送信して, 自 AP 宛に同一
 // Thread ID のメッセージが送信されるまで待ち受ける. 返値は受信したメッ
 // セージ.

 ThreadID tid = agent.send(msg);
 // 非同期メッセージ送信. msg を送信後、直ちにその Thread ID を返却し
 // て処理に戻る.

 agent.setListener(listener);

 // 非同期メッセージ受信のためのリスナ登録. 自 AP 宛のメッセージを受信
 // したら, それを引数として listener.received() を呼び出す (ただし,
 // listener は MessageListener インタフェースを実装するクラスのインス
 // タンスとする).
```

org.t_engine.tenet.core

Class CoreErrorMessageException

```
java.lang.Object
  |-java.lang.Throwable
    |-java.lang.Exception
      |-org.t_engine.tenet.core.CoreException
        |-org.t_engine.tenet.core.CoreErrorMessageException
```

public class **CoreErrorMessageException**
extends CoreException

予期しないメッセージが到達した場合にスローされる。

Constructors

CoreErrorMessageException

```
public CoreErrorMessageException(java.lang.String string)
```

詳細メッセージstringを持つCoreErrorMessageExceptionを構築する。

Parameters:
`string` - **詳細メッセージ**

org.t_engine.tenet.core

Class CoreException

```
java.lang.Object
    |-java.lang.Throwable
        |-java.lang.Exception
            |-org.t_engine.tenet.core.CoreException
```

Direct Known Subclasses:
CoreErrorMessageException, CoreInternalException, CoreParameterException, CoreSmartcardException

---

public class **CoreException**
extends java.lang.Exception

Coreで例外が発生した際にスローされる例外の基底クラスである。

Constructors

CoreException

```
public CoreException(java.lang.String string)
```

詳細メッセージstringを持つCoreExceptionを構築する。

Parameters:
`string` - 詳細メッセージ

org.t_engine.tenet.core

Class CoreInternalException

```
java.lang.Object
    |-java.lang.Throwable
        |-java.lang.Exception
            |-org.t_engine.tenet.core.CoreException
                |-org.t_engine.tenet.core.CoreInternalException
```

public class **CoreInternalException**
extends CoreException

Coreで内部的なエラーが発生した場合にスローされる。
例えば内部リソースファイルアクセスに失敗、設定ファイルに不正、内部のクラス生成に失敗した場合などにスローされる。

Constructors

CoreInternalException

```
public CoreInternalException(java.lang.String string)
```

詳細メッセージstringを持つCoreInternalExceptionを構築する。

Parameters:
string - 詳細メッセージ

org.t_engine.tenet.core

Class CoreParameterException

```
java.lang.Object
  |-java.lang.Throwable
    |-java.lang.Exception
      |-org.t_engine.tenet.core.CoreException
        |-org.t_engine.tenet.core.CoreParameterException
```

---

public class **CoreParameterException**
extends CoreException

メソッドの引数が不正な場合にスローされる。
例えばnullが許容されていないパラメータにnullを指定した場合などにスローされる。

Constructors

CoreParameterException

```
public CoreParameterException(java.lang.String string)
```

詳細メッセージstringを持つCoreParameterExceptionを構築する。

Parameters:
string - 詳細メッセージ

org.t_engine.tenet.core

Class CoreSmartcardException

```
java.lang.Object
   |-java.lang.Throwable
      |-java.lang.Exception
         |-org.t_engine.tenet.core.CoreException
            |-org.t_engine.tenet.core.CoreSmartcardException
```

---

public class **CoreSmartcardException**
extends CoreException

アプリケーションがアクセスするドメインdomainに対するICカードが存在しない場合にスローされる。

Constructors

CoreSmartcardException

`public CoreSmartcardException(java.lang.String string)`

詳細メッセージstringを持つCoreSCardExceptionを構築する。

Parameters:
`string` - 詳細メッセージ

CoreSmartcardException

`public CoreSmartcardException(int errcode)`

エラーコードerrcodeから生成した詳細メッセージを持つCoreSmartcardExceptionを構築する。 エラーコードerrcodeは16進文字列に変換して設定する。

Parameters:
`errcode` - エラーコード

org.t_engine.tenet.core

Class DispatchAgent

```
java.lang.Object
   |
   +-org.t_engine.tenet.core.DispatchAgent
```

public class **DispatchAgent**
extends java.lang.Object

アプリケーションに対してメッセージを送受信するための手段を提供する。
このクラスのインスタンスは、SystemManager#getAgent()メソッドにより取得することができる。それぞれのDispatchAgentクラスのインスタンスは、動的に付与された大域的に一意(global unique)なeTRONIDを識別子として持つ。この識別子は、メッセージ送信の際の送信元ID、およびメッセージ受信の際の宛先IDとして用いられる。付与された識別子の値は、getIdentifier()メソッドにより確認することができる。
アプリケーションは、取得したこのクラスのインスタンスを用いることにより、e2TPメッセージの送受信を行うことができる。メッセージ送受信の方法としては、同期的なメッセージの送受信(メッセージの送信後、送信メッセージに対する応答メッセージの受信を待ち受けする;syncSend()参照)と、非同期的なメッセージの送受信(メッセージの送信後、受信メッセージの待ち受けを行わず直ちにreturnする;send()参照)の、二種類の送受信方法が提供される。
メッセージの送信は、syncSend()もしくはsend()メソッドに対し、送信対象となるMessageクラスのインスタンスを引数として与えることにより行なわれる。ただし、送信されるメッセージの送信元ID(srcID)としては、インスタンス取得の際に付与されたeTRONIDが常に用いられる。すなわち、引数オブジェクトに送信元IDが設定されていたとしても、常に無視される。また、引数オブジェクトのスレッドIDが不定(null)の場合、スレッドIDは自動的に付与される(不定(null)でない場合は、引数オブジェクトに設定されたスレッドIDがそのまま使われる)。この自動付与されるスレッドIDは、大域的な一意性を確保するため、自身 のeTRONIDを含んで生成される(e2TPメッセージ仕様書 2.1.3 節参照)。
非同期にメッセージの受信を行うためには、アプリケーションはあらかじめsetListener()メソッドにより、メッセージの受信を通知するためのリスナオブジェクトを登録する必要がある。登録するリスナオブジェクトは、MessageListenerインタフェースを実装していなくてはならない。メッセージの受信はリスナオブジェクトのMessageListener#received()メソッドの呼び出しとして通知される。リスナオブジェクトが登録されていない場合、受信したメッセージは単に破棄される。

See Also:
Message, MessageListener, SystemManager

Methods

getIdentifier

```
public ETronID getIdentifier()
```

DispatchAgentに割り当てられたeTRONIDを返却する。 このeTRONIDはメッセージ送信時の送信元IDとして自動的に使用される。

Returns:
DispatchAgentに割り当てられたeTRONID

setListener

```
public void setListener(MessageListener listener)
```

リスナlistenerを非同期的なメッセージの受信を行うためのリスナとして登録する。 すでにリスナが登録されている場合は上書きする。リスナlistenerにnullを指定すると登録を解除する。

Parameters:
listener - リスナ

(continued on next page)

(continued from last page)

getListener

```
public MessageListener getListener()
```

setListener()メソッドにより登録されたリスナを取得する。 登録されていない場合はnullを返却する。

Returns:

setListener()メソッドにより登録されたリスナ

---

send

```
public ThreadID send(Message msg)
              throws CoreParameterException
```

送信するメッセージmsgで指定した宛先へ非同期にメッセージを送信する。 メッセージの送信が完了すると直ちに制御を返す。送信するメッセージmsgの送信元IDにはgetIdentifier()メソッドで返却されるeTRONIDを使用(上書き)し、送信するメッセージmsgのスレッドIDが不定(null)の場合は新たに生成したスレッドIDを使用してメッセージを送信する。送信したメッセージのスレッドIDは返値で返却する。

Parameters:

msg - 送信するメッセージ

Returns:

送信したメッセージのスレッドID

Exceptions:

CoreParameterException - 送信するメッセージmsgがnull

---

syncSend

```
public Message syncSend(Message msg)
                throws CoreParameterException
```

送信するメッセージmsgで指定した宛先へ同期的にメッセージを送信する。 送信したメッセージに対応する応答メッセージ、すなわち送信したメッセージと同一スレッドIDを持つメッセージを受信するか、タイムアウトするまで制御を返さない。受信したメッセージは返値で返却し、タイムアウトが発生した場合はnullを返却する。タイムアウト値はSystemManager#getDefaultTimeout()メソッドで返却されるデフォルト値が適用される。send()メソッドと同様に、msgに送信元IDが設定されていたとしても無視される。また、同様にmsgのスレッドIDが不定(null)の場合は自動的にスレッドIDが付与される。

Parameters:

msg - 送信するメッセージ

Returns:

送信したメッセージに対応する応答メッセージ、タイムアウトした場合はnull

Exceptions:

CoreParameterException - 送信するメッセージmsgがnull

---

syncSend

```
public Message syncSend(Message msg,
                        int timeout)
                 throws CoreParameterException
```

送信するメッセージmsgで指定した宛先へメッセージを同期的に送信する(タイムアウト値指定)。 メッセージ受信タイムアウトに引数で与えられたtimeoutが用いられるほかはsyncSend(msg)メソッドと同じである。

Parameters:

msg - 送信するメッセージ
timeout - メッセージ受信タイムアウト(ミリ秒単位の正の整数)

Returns:

(continued from last page)

送信するメッセージに対応する応答メッセージ、タイムアウトした場合はnull

Exceptions:

`CoreParameterException` - 送信するメッセージmsgがnull、またはメッセージ受信タイムアウトtimeoutが負の数

org.t_engine.tenet.core

Class ETronID

```
java.lang.Object
   ┗-org.t_engine.tenet.core.ETronID
```

public class **ETronID**
extends java.lang.Object

ETronIDクラスはe2TPメッセージにおいてメッセージの送信元および宛先を表すための識別子として用いられるeTRONIDを表す。
eTRONIDは、ドメインとポートから構成される。このクラスはeTRONIDのドメインやポートを取得するメソッドなどを提供する。

Constructors

ETronID

```
public ETronID(java.lang.String domain,
               int port)
```

ドメインの文字列表現(24バイトのHex文字列)domainとポートportから構成されるeTRONIDを表す、ETronIDオブジェクトを構築する。

Parameters:
domain - ドメインの文字列表現(24バイトのHex文字列)
port - ポート

Exceptions:
CoreParameterException - ドメインの文字列表現(24バイトのHex文字列)domainがnull、または24バイトでない、またはHex文字列でない

ETronID

```
public ETronID(byte[] domain,
               int port)
```

ドメインのバイト配列表現domainとポートportから構成されるeTRONIDを表す、ETronIDオブジェクトを構築する。

Parameters:
domain - ドメインのバイト配列表現
port - ポート

Exceptions:
CoreParameterException - ドメインのバイト配列表現domainがnullか12バイト以外

ETronID

```
public ETronID(java.lang.String eTRONID)
```

eTRONIDの文字列表現(32バイトのHex文字列)eTRONIDで示されるeTRONIDを表す、ETronIDオブジェクトを構築する。

Parameters:
eTRONID - eTRONIDの文字列表現(32バイトのHex文字列)

(continued from last page)

Exceptions:
CoreParameterException - eTRONIDの文字列表現(32バイトのHex文字列)eTRONIDがnullか、または32バイト以外か、またはHex文字列ではない

ETronID

```
public ETronID(byte[] eTRONID)
```

eTRONIDのバイト配列表現eTRONIDで示されるeTRONIDを表す、ETronIDオブジェクトを構築する。

Parameters:
eTRONID - eTRONIDのバイト配列表現

Exceptions:
CoreParameterException - eTRONIDのバイト配列表現eTRONIDがnullか16バイトでない

Methods

equals

```
public boolean equals(java.lang.Object obj)
```

eTRONIDの等値性を判定する。 比較対象objが表すeTRONIDが、このオブジェクトのeTRONIDと等しい場合はtrueが返却される。eTRONIDが異なる場合、もしくは比較対象objがETronIDクラスのインスタンスでない場合はfalseが返却される。

Parameters:
obj - 比較対象

Returns:
比較対象objが表すeTRONIDがこのオブジェクトのeTRONIDと等しい場合はtrue、そうでない場合はfalse

---

toString

```
public java.lang.String toString()
```

このeTRONIDの文字列表現(32バイトのHex文字列)を返す。

Returns:
eTRONIDの文字列表現(32バイトのHex文字列)

---

toBytes

```
public byte[] toBytes()
```

このeTRONIDのバイト配列表現を返す。

Returns:
eTRONIDのバイト配列表現

---

getDomain

```
public java.lang.String getDomain()
```

このeTRONIDのドメインの文字列表現(24バイト文字列)を返す。

Returns:
ドメインの文字列表現(24バイト文字列)

(continued from last page)

getPort

```
public int getPort()
```

このeTRONIDのポートを返す。

Returns:

ポート

org.t_engine.tenet.core

Class Message

```
java.lang.Object
   ┗-org.t_engine.tenet.core.Message
```

public class **Message**
extends java.lang.Object

Messageクラスはe2TPメッセージを表す。
このオブジェクトには、メッセージを送信したアプリケーションまたはICカードのeTRONIDである送信元ID(src)、メッセージの宛先となるアプリケーションまたはICカードのeTRONIDである宛先ID(dest)、メッセージの種別ごとに定められたメッセージ種別コード(mtype)、一連のメッセージ送受信を識別するためのスレッドID(threadID)、メッセージ種別ごとに定められたフォーマットに従って記述されたパラメータ(param)が含まれる。
このクラスのコンストラクタでは、メッセージの送信元IDを陽に引数として指定することはできない。すなわち、コンストラクタにより生成されたMessageオブジェクトの送信元IDは不定である。また、コンストラクタ(dest、mtype、param)により生成されたMessageオブジェクトのスレッドIDは不定である。
このクラスは送信元ID、宛先ID、メッセージ種別コード、スレッドID、パラメータを取得するメソッドや、メッセージのバイト配列表現を返却するメソッドを提供する。

Constructors

Message

```
public Message(ETronID dest,
               int mtype,
               byte[] param)
```

宛先ID dest、メッセージ種別コードmtype、パラメータparamから構成されるメッセージを表す、Messageオブジェクトを構築する。 このコンストラクタにより生成されたMessageオブジェクトはスレッドIDが不定となる。パラメータparamが存在しない場合はnullを指定する。

Parameters:
dest - 宛先ID
mtype - メッセージ種別コード
param - パラメータ

Exceptions:
CoreParameterException - 宛先ID destがnull

Message

```
public Message(ETronID dest,
               int mtype,
               ThreadID threadID,
               byte[] param)
```

宛先ID dest、メッセージ種別コードmtype スレッドID threadID、パラメータparamから構成されるメッセージを表す、Messageオブジェクトを構築する。 パラメータparamが存在しない場合はnullを指定する。

Parameters:
dest - 宛先ID
mtype - メッセージ種別コード
threadID - スレッドID
param - パラメータ

Exceptions:

(continued from last page)

CoreParameterException - 宛先ID destがnull

Methods

toBytes

public byte[] **toBytes**()

このメッセージのバイト配列表現を返す。

Returns:

メッセージのバイト配列表現

---

getSrc

public ETronID **getSrc**()

このメッセージの送信元IDを返す。 送信元IDが不定の場合はnullを返却する。

Returns:

送信元ID

---

getDest

public ETronID **getDest**()

このメッセージの宛先IDを返す。

Returns:

宛先ID

---

getMtype

public int **getMtype**()

このメッセージのメッセージ種別コードを返す。

Returns:

メッセージ種別コード

---

getThreadID

public ThreadID **getThreadID**()

このメッセージのスレッドIDを返す。 スレッドIDが不定の場合はnullを返却する。

Returns:

スレッドID

---

getParam

public byte[] **getParam**()

このメッセージのパラメータを返す。 パラメータが存在しない場合は0バイトの配列を返却する。

Returns:

パラメータ

org.t_engine.tenet.core

Interface MessageListener

public interface **MessageListener**

MessageListenerインタフェースは非同期的なメッセージの送信に対する応答メッセージを受信をするための、リスナを構成するためのインタフェースである。
このインタフェースを実装したリスナをDispatchAgent#setListener()メソッドにて登録することでメッセージ受信の通知を適宜受けることができる。

See Also:
DispatchAgent

Methods

received

```
public Message received(Message msg)
```

このオブジェクトを登録したDispatchAgentオブジェクトがメッセージを受信したときに呼び出される。引数msgは受信したメッセージである。このメソッドの返値をMessageクラスのオブジェクトとすることにより、返値オブジェクトはメッセージとして自動的に送信される。このとき、送信元IDとしてこのオブジェクトを登録したDispatchAgentオブジェクトのeTRONIDが、スレッドIDが不定(null)の場合は受信メッセージと同一のスレッドIDが、それぞれ送信前に自動的に設定される。 返値をnullもしくはMessageクラス以外のオブジェクトとした場合は、メッセージは自動的には送信されない。メッセージの自動送信が必要ない場合、返値としてnullを返却することが推奨される。

Parameters:
msg - 受信したメッセージ

Returns:
対応して送信するメッセージ、不要な場合はnull

org.t_engine.tenet.core

Class SystemManager

```
java.lang.Object
   |
   +-org.t_engine.tenet.core.SystemManager
```

public class **SystemManager**
extends java.lang.Object

このクラスはシステムを管理する、VM(または仮想VM)上でsingletonなクラスである。
接続されているICカードリーダライタ(ICカードR/W)の名称とICカードR/Wに挿入されたICカードのドメインのMap、同期的メッセージ送受信におけるデフォルトタイムアウト値を返却するメソッドや、アプリケーションに対してメッセージを送受信する機能を提供するDispatchAgentを返却するメソッドを提供する。

See Also:
DispatchAgent

Methods

getInstance

```
public static SystemManager getInstance()
                            throws CoreInternalException
```

本APIを用いてe2TPメッセージを送受するために必要な環境の初期化を行い、このクラスの唯一のインスタンスを返す。

Returns:
唯一のインスタンス

Exceptions:
`CoreInternalException` - リソース(ICカードR/Wドライバ)へのアクセスに失敗

getDomainMap

```
public org.t_engine.tenet.util.Map getDomainMap()
```

ICカードR/W名称をキーとした、ICカードR/Wに挿入されているICカードのドメインの文字列表現(24バイトのHex文字列)domainのMapを返す。 ICカードR/Wに挿入されている使用可能なICカードが環境に存在しない場合は空のMapを返却する。

Returns:
ICカードR/W名称をキーとした、ICカードR/Wに挿入されているICカードのドメインの文字列表現(24バイトのHex文字列)のMap

getDefaultTimeout

```
public int getDefaultTimeout()
```

同期的なメッセージ送受信におけるデフォルトのタイムアウトを返す。 デフォルトのタイムアウト値は環境に依存する。

Returns:
デフォルトのタイムアウト(ミリ秒単位の正の整数)

(continued from last page)

setDefaultTimeout

```
public void setDefaultTimeout(int timeout)
                       throws CoreParameterException
```

同期的なメッセージ送受信におけるデフォルトのタイムアウトを設定する。

Parameters:
timeout - デフォルトのタイムアウト(ミリ秒単位の正の整数)

Exceptions:
CoreParameterException - デフォルトのタイムアウトtimeoutが負の数

---

getAgent

```
public DispatchAgent getAgent(java.lang.String domain)
                       throws CoreParameterException,
                              CoreErrorMessageException,
                              CoreSmartcardException
```

ドメインdomainに属するeTRONIDを持ったDispatchAgentオブジェクトを新たに生成し、返却する。

Parameters:
domain - ドメイン

Returns:
生成したDispatchAgentオブジェクト

Exceptions:
CoreParameterException - 指定されたdomainが正しいドメインの文字列表現(see ETronID)ではない
CoreErrorMessageException - 生成するDispatchAgentオブジェクトのためのポートをICカードが生成することに失敗した(ドメインdomainはこれ以上新たなポートを生成できない)
CoreSmartcardException - 指定されたdomainに対応するICカードが環境に存在しない

org.t_engine.tenet.core

Class ThreadID

```
java.lang.Object
   |
   +-org.t_engine.tenet.core.ThreadID
```

---

public class **ThreadID**
extends java.lang.Object

ThreadIDクラスはメッセージに含まれるスレッドIDを表す。
このクラスはスレッドIDの文字列表現(40バイトのHex文字列)やバイト配列表現を返却するメソッドを提供する。

Constructors

ThreadID

```
public ThreadID(byte[] threadID)
```

ThreadIDのバイト列表現threadIDで示されるThreadIDを表す、ThreadIDオブジェクトを構築する。

Parameters:
`threadID` - ThreadIDのバイト列表現

Exceptions:
`CoreParameterException` - threadIDがnullか20バイトでない

Methods

toString

```
public java.lang.String toString()
```

このスレッドIDの文字列表現(40バイトのHex文字列)を返す。

Returns:
スレッドIDの文字列表現(40バイトのHex文字列)

---

toBytes

```
public byte[] toBytes()
```

このスレッドIDのバイト配列表現を返す。

Returns:
スレッドIDのバイト配列表現

Index

Appendix org.t_engine.util.* について

権利価値取引 API, e^{2}TP メッセージング API は, いずれも集合および連想配列を扱うためのインタフェース, およびイテレータを扱うためのインタフェースとして, org.t_engine.util.Set, org.t_engine.util.Map, org.t_engine.util.Iterator の 3 種類のインタフェースを利用している.

これらのインタフェースは, それぞれ java.util パッケージに存在する同名のインタフェース (java.util.Set, java.util.Map, java.util.Iterator) と, 以下の例外を除いて同一のインタフェースを提供するものとする.

すなわち, java.util.Collection が引数もしくは返値として現れるメソッドについては, 代わりに org.t_engine.util.Collection を用いるものとする.
org.t_engine.util.Collection インタフェースは, java.util.Collection インタフェースと同一のインタフェースを提供する.

これは, J2ME CLDC 環境では JCF (Java Collections Framework) が利用できないことによる対処である.